Applications Area Working Group
Internet-Draft
Updates: 1738 (if approved)
Intended status: Standards Track
Expires: June 19, 2017

M. Kerwin
QUT

December 16, 2016

The file URI Scheme
draft-ietf-appsawg-file-scheme-16

Abstract

This document provides a more complete specification of the "file" Uniform Resource Identifier (URI) scheme, replacing the very brief definition in Section 3.10 of RFC 1738.

It defines a common syntax which is intended to interoperate across the broad spectrum of existing usages. At the same time it notes some other current practices around the use of file URIs.

Note to Readers (To be removed by the RFC Editor)

This draft should be discussed on the IETF Applications Area Working Group discussion list <apps-discuss@ietf.org>.

Table of Contents

1. Introduction

A file URI identifies an object (a "file") stored in a structured object naming and accessing environment on a host (a "file system.") The URI can be used in discussions about the file, and if other

conditions are met it can be dereferenced to directly access the file.

This document specifies a syntax based on the generic syntax of [RFC3986] that is compatible with most existing usages. Where incompatibilities arise they are usually in parts of the scheme that were underspecified in earlier definitions and have been tightened up by more recent specifications. Appendix A lists significant changes to syntax.

Extensions to the syntax which might be encountered in practice are listed in Appendix E; these extensions are listed for informational purposes and are not a requirement of implementation.

The file URI scheme is not coupled with a specific protocol, nor with a specific media type [RFC6838]. See Section 3 for a discussion of operations that can be performed on the object identified by a file URI.

1.1. 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] when they appear in all upper case. They may also appear in lower or mixed case as English words, without normative meaning.

Throughout this document the term "local file" is used to describe files that can be accessed through the local file system API using only the information included in the file path, not relying on other information (such as network addresses.) It is important to note that a local file may not be physically located on the local machine, for example if a networked file system is transparently mounted into the local file system.

The term "local file URI" is used to describe file URIs which have no "authority" component, or where the authority is the special string "localhost" or a fully qualified domain name that resolves to the machine from which the URI is being interpreted (Section 2).

2. Syntax

The file URI syntax is defined here in Augmented Backus-Naur Form (ABNF) [RFC5234], importing the "host" and "path-absolute" rules from [RFC3986] (as updated by [RFC6874].)

The generic syntax in [RFC3986] includes "path" and "authority" components, for each of which only a subset is used in the definition

of the file URI scheme. The relevant subset of "path" is "path-absolute", and the subset of "authority" is "file-auth", given below.

The syntax definition below is different from those given in [RFC1630] and [RFC1738] as it is derived from the generic syntax of [RFC3986], which post-dates the previous file URI specifications. Appendix A enumerates significant differences.

```
file-URI       = file-scheme ":" file-hier-part

file-scheme    = "file"

file-hier-part = ( "//" auth-path )
               / local-path

auth-path      = [ file-auth ] path-absolute

local-path     = path-absolute

file-auth      = "localhost"
               / host
```

The "host" is the fully qualified domain name of the system on which the file is accessible. This allows a client on another system to know that it cannot access the file system, or perhaps that it needs to use some other local mechanism to access the file.

As a special case, the "file-auth" rule can match the string "localhost" which is interpreted as "the machine from which the URI is being interpreted," exactly as if no authority were present. Some current usages of the scheme incorrectly interpret all values in the authority of a file URI, including "localhost", as non-local. Yet others interpret any value as local, even if the "host" does not resolve to the local machine. To maximize compatibility with previous specifications, users MAY choose to include an "auth-path" with no "file-auth" when creating a URI.

The path component represents the absolute path to the file in the file system. See Appendix D for some discussion of system-specific concerns including absolute file paths and file system roots.

Some file systems have case-sensitive file naming and some do not. As such the file URI scheme supports case sensitivity, in order to retain the case as given. Any transport-related handling of the file URI scheme MUST retain the case as given. Any mapping to or from a case-insensitive form is solely the responsibility of the implementation processing the file URI on behalf of the referenced file system.

Also see Appendix E that lists some nonstandard syntax variations that can be encountered in practice.

3. Operations Involving file URIs

See the POSIX file and directory operations [POSIX] for examples of standardized operations that can be performed on files.

A file URI can be dependably dereferenced or translated to a local file path only if it is local. A file URI is considered "local" if it has no "file-auth", or the "file-auth" is the special string "localhost" or a fully qualified domain name that resolves to the machine from which the URI is being interpreted (Section 2).

This specification neither defines nor forbids any set of operations that might be performed on a file identified by a non-local file URI.

4. File System Name Encoding

File systems use various encoding schemes to store file and directory names. Many modern file systems store file and directory names as arbitrary sequences of octets, in which case the representation as an encoded string often depends on the user's localization settings, or defaults to UTF-8 [STD63].

When a file URI is produced that represents textual data consisting of characters from the Unicode Standard coded character set [UNICODE], the data SHOULD be encoded as octets according to the UTF-8 character encoding scheme [STD63] before percent-encoding is applied; as per [RFC3986], Section 2.5.

A decision not to use percent-encoded UTF-8 is outside the scope of this specification. It will typically require the use of heuristics or explicit knowledge about the way the string will be processed.

5. Security Considerations

There are many security considerations for URI schemes discussed in [RFC3986].

File access and the granting of privileges for specific operations are complex topics, and the use of file URIs can complicate the security model in effect for file privileges.

Historically, user agents have granted content from the file URI scheme a tremendous amount of privilege. However, granting all local files such wide privileges can lead to privilege escalation attacks. Some user agents have had success granting local files directory-

based privileges, but this approach has not been widely adopted. Other user agents use globally unique identifiers as the origin for each file URI [RFC6454], which is the most secure option.

Treating a non-local file URI as local or otherwise attempting to perform local operations on a non-local URI can result in security problems.

File systems typically assign an operational meaning to special characters, such as the "/", "\", ":", "[", and "]" characters, and to special device names like ".", "..", "...", "aux", "lpt", etc. In some cases, merely testing for the existence of such a name will cause the operating system to pause or invoke unrelated system calls, leading to significant security concerns regarding denial of service and unintended data transfer. It would not be possible for this specification to list all such significant characters and device names. Implementers should research the reserved names and characters for the types of storage device that may be attached to their application and restrict the use of data obtained from URI components accordingly.

File systems vary in the way they handle case. Care must be taken to avoid issues resulting from possibly unexpected aliasing from case-only differences between file paths or URIs, or from mismatched encodings or Unicode equivalences [UAX15] (see Section 4).

6. IANA Considerations

This document defines the following URI scheme, so the "Permanent URI Schemes" registry has been updated accordingly. This registration complies with [BCP35].

Scheme name:
file

Status:
permanent

Applications/protocols that use this scheme name:
Commonly used in hypertext documents to refer to files without depending on network access. Supported by major browsers.

Used in development libraries, such as:

* Windows Shell (PathCreateFromUrl, UrlCreateFromPath).

* libwww-perl - The World-Wide Web library for Perl.

Contact:
Applications and Real-Time Area <art@ietf.org>

Change Controller:
This scheme is registered under the IETF tree. As such, the IETF maintains change control.

References:
This RFC.

7. Acknowledgements

Contributions from many members of the IETF and W3C communities - notably Dave Crocker, Graham Klyne, Tom Petch, and John Klensin - are greatly appreciated.

Additional thanks to Dave Risney, author of the informative IE Blog article <http://blogs.msdn.com/b/ie/archive/2006/12/06/file-uris-in-windows.aspx>, and Dave Thaler for their early comments and suggestions; and to Paul Hoffman, whose earlier work served as an inspiration for this undertaking.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <http://www.rfc-editor.org/info/rfc6454>.

[RFC6874] Carpenter, B., Cheshire, S., and R. Hinden, "Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers", RFC 6874, DOI 10.17487/RFC6874, February 2013, <http://www.rfc-editor.org/info/rfc6874>.

[STD63] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/std63>.

8.2. Informative References

[Bash-Tilde]
Free Software Foundation, Inc, "Bash Reference Manual: Tilde Expansion", February 2014, <http://www.gnu.org/software/bash/manual/html_node/Tilde-Expansion.html>.

[BCP35] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <http://www.rfc-editor.org/info/bcp35>.

[Bug107540]
Bugzilla@Mozilla, "Bug 107540", October 2007, <https://bugzilla.mozilla.org/show_bug.cgi?id=107540>.

[MS-DTYP] Microsoft Open Specifications, "Windows Data Types, 2.2.57 UNC", October 2015, <http://msdn.microsoft.com/en-us/library/gg465305.aspx>.

[POSIX] IEEE, "IEEE Std 1003.1, 2013 Edition", 2013.

[RFC1630] Berners-Lee, T., "Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web", RFC 1630, DOI 10.17487/RFC1630, June 1994, <http://www.rfc-editor.org/info/rfc1630>.

[RFC1738] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, DOI 10.17487/RFC1738, December 1994, <http://www.rfc-editor.org/info/rfc1738>.

[RFC2396] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, DOI 10.17487/RFC2396, August 1998, <http://www.rfc-editor.org/info/rfc2396>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <http://www.rfc-editor.org/info/rfc6838>.

[UAX15] Davis, M., Ed. and K. Whistler, Ed., "Unicode Standard Annex #15: Unicode Normalization Forms", February 2016, <http://www.unicode.org/reports/tr15/tr15-44.html>.

[UNICODE] The Unicode Consortium, "The Unicode Standard, Version 9.0.0", ISBN 978-1-936213-13-9, June 2016, <http://www.unicode.org/versions/Unicode9.0.0/>.

[WHATWG-URL] WHATWG, "URL Standard", December 2016, <https://url.spec.whatwg.org/>.

[Win32-Namespaces] Microsoft Developer Network, "Naming Files, Paths, and Namespaces", June 2013, <https://msdn.microsoft.com/en-us/library/windows/desktop/aa365247(v=vs.85).aspx>.

[Zsh-Tilde] "zsh: 14.7 Filename Expansion", December 2015, <http://zsh.sourceforge.net/Doc/Release/Expansion.html#Filename-Expansion>.

Appendix A. Differences from Previous Specifications

The syntax definition in Section 2 inherits incremental differences from the general syntax of [RFC1738] made by [RFC2396] ([RFC2396], Appendix G) and [RFC3986] ([RFC3986], Appendix D).

According to the definition in [RFC1738] a file URL always started with the token "file://", followed by an (optionally blank) host name and a "/". The syntax given in Section 2 makes the entire authority component, including the double slashes "//", optional.

Appendix B. Example URIs

The syntax in Section 2 is intended to support file URIs that take the following forms:

Local files:

- A traditional file URI for a local file, with an empty authority. This is the most common format in use today. E.g.:

    * "file:///path/to/file"

- The minimal representation of a local file, with no authority field and an absolute path that begins with a slash "/". E.g.:

    * "file:/path/to/file"

Non-local files:

- A non-local file, with an explicit authority. E.g.:

    * "file://host.example.com/path/to/file"

Appendix C. Similar Technologies

- The WHATWG URL specification [WHATWG-URL] defines browser behavior for a variety of inputs, including file URIs. As a living document, it changes to reflect updates in browser behavior. As a result, its algorithms and syntax definitions may or may not be consistent with this specification. Implementors should be aware of this possible discrepancy if they expect to share file URIs with browsers that follow the WHATWG specification.

- The Universal Naming Convention (UNC) [MS-DTYP] defines a string format that can perform a similar role to the file URI scheme in describing the location of files, except that files located by UNC filespace selector strings are typically stored on a remote machine and accessed using a network protocol. Appendix E.3 lists some ways in which UNC filespace selector strings are currently made to interoperate with the file URI scheme.

- The Microsoft Windows API defines Win32 Namespaces [Win32-Namespaces] for interacting with files and devices using Windows API functions. These namespaced paths are prefixed by "\\?\" for Win32 File Namespaces and "\\.\" for Win32 Device Namespaces. There is also a special case for UNC file paths in Win32 File Namespaces, referred to as "Long UNC", using the prefix "\\?\UNC\". This specification does not define a mechanism for translating namespaced paths to or from file URIs.

Appendix D. System-Specific Operations

This appendix is not normative. It highlights some observed behaviours and provides system-specific guidance for interacting with file URIs and paths. This is not an exhaustive list of operating or file systems; rather it is intended to illustrate certain types of interactions that might be encountered.

D.1. POSIX Systems

In a POSIX file system the root of the file system is represented as a directory with a zero-length name, usually written as "/"; the presence of this root in a file URI can be taken as given by the initial slash in the "path-absolute" rule.

Common UNIX shells such as the Bourne-Again SHell (bash) and Z Shell (zsh) provide a function known as "tilde expansion" [Bash-Tilde] or "filename expansion" [Zsh-Tilde], where a path that begins with a tilde character "~" can be expanded out to a special directory name. No such facility exists using the file URI scheme; a tilde in a file URI is always just a tilde.

D.2. DOS- and Windows-Like Systems

When mapping a DOS- or Windows-like file path to a file URI, the drive letter (e.g. "c:") is typically mapped into the first path segment.

Appendix E lists some nonstandard techniques for interacting with DOS- or Windows-like file paths and URIs.

D.3. Mac OS X Systems

The HFS+ file system uses a nonstandard normalization form, similar to Normalization Form D [UAX15]. Take care when transforming HFS+ file paths to and from URIs (Section 4).

D.4. OpenVMS Files-11 Systems

When mapping a VMS file path to a file URI, the device name is mapped into the first path segment. Note that the dollars sign "$" is a reserved character per the definition in [RFC3986], Section 2.2, so should be percent-encoded if present in the device name.

If the VMS file path includes a node reference, that reference is used as the authority. Where the original node reference includes a user name and password in an access control string, they can be transcribed into the authority using the nonstandard syntax extension in Appendix E.1.

Appendix E. Nonstandard Syntax Variations

These variations may be encountered by existing usages of the file URI scheme, but are not supported by the normative syntax of Section 2.

This appendix is not normative.

E.1. User Information

It might be necessary to include user information such as a user name in a file URI, for example when mapping a VMS file path with a node reference that includes an access control string.

To allow user information to be included in a file URI, the "file-auth" rule in Section 2 can be replaced with the following:

```
file-auth      = "localhost"
               / [ userinfo "@" ] host
```

This uses the "userinfo" rule from [RFC3986].

As discussed in the HP OpenVMS Systems Documentation <http://h71000.www7.hp.com/doc/84final/ba554_90015/ch03s09.html> "access control strings include sufficient information to allow someone to break in to the remote account, [therefore] they create serious security exposure." In a similar vein, the presence of a password in a "user:password" userinfo field is deprecated by [RFC3986]. Take care when dealing with information that can be used to identify a user or grant access to a system.

E.2. DOS and Windows Drive Letters

On Windows- or DOS-like file systems an absolute file path can begin with a drive letter. To facilitate this, the "local-path" rule in Section 2 can be replaced with the following:

```
local-path     = [ drive-letter ] path-absolute

drive-letter   = ALPHA ":"
```

The "ALPHA" rule is defined in [RFC5234].

This is intended to support the minimal representation of a local file in a DOS- or Windows-like environment, with no authority field and an absolute path that begins with a drive letter. E.g.:

o "file:c:/path/to/file"

URIs of the form "file:///c:/path/to/file" are already supported by the "path-absolute" rule.

Note that comparison of drive letters in DOS or Windows file paths is case-insensitive. In some usages of file URIs drive letters are

canonicalized by converting them to uppercase, and other usages treat URIs that differ only in the case of the drive letter as identical.

E.2.1. Relative Resolution

To mimic the behaviour of DOS- or Windows-like file systems, relative references beginning with a slash "/" can be resolved relative to the drive letter, when present; and resolution of ".." dot segments (per Section 5.2.4 of [RFC3986]) can be modified to not ever overwrite the drive letter.

For example:

```
base URI:   file:///c:/path/to/file.txt
rel. ref.:  /some/other/thing.bmp
resolved:   file:///c:/some/other/thing.bmp

base URI:   file:///c:/foo.txt
rel. ref.:  ../bar.txt
resolved:   file:///c:/bar.txt
```

A relative reference starting with a drive letter would be interpreted by a generic URI parser as a URI with the drive letter as its scheme. Instead such a reference ought to be constructed with a leading slash "/" character (e.g. "/c:/foo.txt").

Relative references with a drive letter followed by a character other than a slash (e.g. "/c:bar/baz.txt" or "/c:../foo.txt") might not be accepted as dereferenceable URIs in DOS- or Windows-like systems.

E.2.2. Vertical Line Character

Historically some usages of file URIs have included a vertical line character "|" instead of a colon ":" in the drive letter construct. [RFC3986] forbids the use of the vertical line, however it may be necessary to interpret or update old URIs.

For interpreting such URIs, the "auth-path" and "local-path" rules in Section 2 and the "drive-letter" rule above can be replaced with the following:

```
auth-path      = [ file-auth ] path-absolute
               / [ file-auth ] file-absolute

local-path     = [ drive-letter ] path-absolute
               / file-absolute

file-absolute  = "/" drive-letter path-absolute

drive-letter   = ALPHA ":"
               / ALPHA "|"
```

This is intended to support regular DOS or Windows file URIs with vertical line characters in the drive letter construct. E.g.:

- "file:///c|/path/to/file"
- "file:/c|/path/to/file"
- "file:c|/path/to/file"

To update such an old URI, replace the vertical line "|" with a colon ":".

E.3. UNC Strings

Some usages of the file URI scheme allow UNC filespace selector strings [MS-DTYP] to be translated to and from file URIs, either by mapping the equivalent segments of the two schemes (hostname to authority, sharename+objectnames to path), or by mapping the entire UNC string to the path segment of a URI.

E.3.1. file URI with Authority

The following is an algorithmic description of the process of translating a UNC filespace selector string to a file URI by mapping the equivalent segments of the two schemes:

1. Initialize the URI with the "file:" scheme identifier.
2. Append the authority:
    1. Append the "//" authority sigil to the URI.
    2. Append the host-name field of the UNC string to the URI.
3. Append the share-name:

1. Transform the share-name to a path segment ([RFC3986], Section 3.3) to conform to the encoding rules of Section 2 of [RFC3986].

2. Append a delimiting slash character "/" and the transformed segment to the URI.

4. For each object-name:

1. Transform the objectname to a path segment as above.

The colon character ":" is allowed as a delimiter before stream-name and stream-type in the file-name, if present.

2. Append a delimiting slash character "/" and the transformed segment to the URI.

For example:

```
UNC String:   \\host.example.com\Share\path\to\file.txt
URI:          file://host.example.com/Share/path/to/file.txt
```

The inverse algorithm, for translating a file URI to a UNC filespace selector string, is left as an exercise for the reader.

E.3.2. file URI with UNC Path

It is common to encounter file URIs that encode entire UNC strings in the path, usually with all backslash "\" characters replaced with slashes "/".

To interpret such URIs, the "auth-path" rule in Section 2 can be replaced with the following:

```
auth-path      = [ file-auth ] path-absolute
               / unc-authority path-absolute

unc-authority  = 2*3"/" file-host

file-host      = inline-IP / IPv4address / reg-name

inline-IP      = "%5B" ( IPv6address / IPvFuture ) "%5D"
```

This syntax uses the "IPv4address", "IPv6address", "IPvFuture", and "reg-name" rules from [RFC3986].

Note that the "file-host" rule is the same as "host" but with percent-encoding applied to "[" and "]" characters.

This extended syntax is intended to support URIs that take the following forms, in addition to those in Appendix B:

Non-local files:

- The representation of a non-local file, with an empty authority and a complete (transformed) UNC string in the path. E.g.:
  - "file:////host.example.com/path/to/file"
- As above, with an extra slash between the empty authority and the transformed UNC string, as per the syntax defined in [RFC1738]. E.g.:
  - "file://///host.example.com/path/to/file"

  This representation is notably used by the Firefox web browser. See Bugzilla#107540 [Bug107540].

It also further limits the definition of a "local file URI" by excluding any file URI with a path that encodes a UNC string.

E.4. Backslash as Separator

Historically some usages have copied entire file paths into the path components of file URIs. Where DOS or Windows file paths were thus copied the resulting URI strings contained unencoded backslash "\" characters, which are forbidden by both [RFC1738] and [RFC3986].

It may be possible to translate or update such an invalid file URI by replacing all backslashes "\" with slashes "/", if it can be determined with reasonable certainty that the backslashes are intended as path separators.

Appendix F. Collected Nonstandard Rules

Here are the collected syntax rules for all optional appendices, presented for convenience. This collected syntax is not normative.

```
file-URI       = file-scheme ":" file-hier-part

file-scheme    = "file"

file-hier-part = ( "//" auth-path )
               / local-path

auth-path      = [ file-auth ] path-absolute
               / [ file-auth ] file-absolute
               / unc-authority path-absolute

local-path     = [ drive-letter ] path-absolute
               / file-absolute

file-auth      = "localhost"
               / [ userinfo "@" ] host

unc-authority  = 2*3"/" file-host

file-host      = inline-IP / IPv4address / reg-name

inline-IP      = "%5B" ( IPv6address / IPvFuture ) "%5D"

file-absolute  = "/" drive-letter path-absolute

drive-letter   = ALPHA ":"
               / ALPHA "|"
```

This collected syntax is intended to support file URIs that take the following forms:

Local files:

- A traditional file URI for a local file, with an empty authority. E.g.:
  - "file:///path/to/file"
- The minimal representation of a local file, with no authority field and an absolute path that begins with a slash "/". E.g.:
  - "file:/path/to/file"
- The minimal representation of a local file in a DOS- or Windows-based environment, with no authority field and an absolute path that begins with a drive letter. E.g.:
  - "file:c:/path/to/file"

- Regular DOS or Windows file URIs, with vertical line characters in the drive letter construct. E.g.:
  - "file:///c|/path/to/file"
  - "file:/c|/path/to/file"
  - "file:c|/path/to/file"

Non-local files:

- The representation of a non-local file, with an explicit authority. E.g.:
  - "file://host.example.com/path/to/file"
- The "traditional" representation of a non-local file, with an empty authority and a complete (transformed) UNC string in the path. E.g.:
  - "file:////host.example.com/path/to/file"
- As above, with an extra slash between the empty authority and the transformed UNC string. E.g.:
  - "file://///host.example.com/path/to/file"

Author's Address

Matthew Kerwin
Queensland University of Technology
Victoria Park Road
Kelvin Grove, QLD 4059
Australia

Email: matthew.kerwin@qut.edu.au

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 23, 2016

M. Nottingham
Akamai
E. Wilde

January 20, 2016

Problem Details for HTTP APIs
draft-ietf-appsawg-http-problem-03

Abstract

This document defines a "problem detail" as a way to carry machine-readable details of errors in a HTTP response, to avoid the need to define new error response formats for HTTP APIs.

Note to Readers

This draft should be discussed on the apps-discuss mailing list [1].

This section is to be removed before publication.

Note to RFC Editor

Please replace all occurrences of "XXXX" with the final RFC number chosen for this draft.

This section is to be removed before publication.

Status of This Memo

Table of Contents

1. Introduction

HTTP [RFC7230] status codes are sometimes not sufficient to convey enough information about an error to be helpful. While humans behind Web browsers can be informed about the nature of the problem with an HTML [W3C.REC-html5-20141028] response body, non-human consumers of so-called "HTTP APIs" are usually not.

This specification defines simple JSON [RFC7159] and XML [W3C.REC-xml-20081126] document formats to suit this purpose. They are designed to be reused by HTTP APIs, which can identify distinct "problem types" specific to their needs.

Thus, API clients can be informed of both the high-level error class (using the status code) and the finer-grained details of the problem (using one of these formats).

For example, consider a response that indicates that the client's account doesn't have enough credit. The 403 Forbidden status code might be deemed most appropriate to use, as it will inform HTTP-generic software (such as client libraries, caches and proxies) of the general semantics of the response.

However, that doesn't give the API client enough information about why the request was forbidden, the applicable account balance, or how to correct the problem. If these details are included in the response body in a machine-readable format, the client can treat it appropriately; for example, triggering a transfer of more credit into the account.

This specification does this by identifying a specific type of problem (e.g., "out of credit") with a URI [RFC3986]; HTTP APIs can do this by nominating new URIs under their control, or by reusing existing ones.

Additionally, problems can contain other information, such as a URI that identifies the specific occurrence of the problem (effectively giving an identifier to the concept "The time Joe didn't have enough credit last Thursday"), which can be useful for support or forensic purposes.

The data model for problem details is a JSON [RFC7159] object; when formatted as a JSON document, it uses the "application/problem+json" media type. Appendix A defines how to express them in an equivalent XML format, which uses the "application/problem+xml" media type.

Note that problem details are (naturally) not the only way to convey the details of a problem in HTTP; if the response is still a representation of a resource, for example, it's often preferable to accommodate describing the relevant details in that application's format. Likewise, in many situations, there is an appropriate HTTP status code that does not require extra detail to be conveyed.

Instead, the aim of this specification is to define common error formats for those applications that need one, so that they aren't required to define their own, or worse, tempted to re-define the semantics of existing HTTP status codes. Even if an application chooses not to use it to convey errors, reviewing its design can help guide the design decisions faced when conveying errors in an existing format.

2. Requirements

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. The Problem Details JSON Object

The canonical model for problem details is a JSON [RFC7159] object.

When serialised as a JSON document, that format is identified with the "application/problem+json" media type.

For example, a HTTP response carrying JSON problem details:

```
HTTP/1.1 403 Forbidden
Content-Type: application/problem+json
Content-Language: en

{
 "type": "https://example.com/probs/out-of-credit",
 "title": "You do not have enough credit.",
 "detail": "Your current balance is 30, but that costs 50.",
 "instance": "/account/12345/msgs/abc",
 "balance": 30,
 "accounts": ["/account/12345",
              "/account/67890"]
}
```

Here, the out-of-credit problem (identified by its type URI) indicates the reason for the 403 in "title", gives a reference for the specific problem occurrence with "instance", gives occurrence-specific details in "detail", and adds two extensions; "balance" conveys the account’s balance, and "accounts" gives links where the account can be topped up.

The ability to convey problem-specific extensions allows more than one problem to be conveyed. For example:

```
HTTP/1.1 400 Bad Request
Content-Type: application/problem+json
Content-Language: en

{
"type": "https://example.net/validation-error",
"title": "Your request parameters didn't validate.",
"invalid-params": [ {
                      "name": "age",
                      "reason": "must be a positive integer"
                    },
                    {
                      "name": "color",
                      "reason": "must be 'green', 'red' or 'blue'"}
                  ]
}
```

Note that this requires each of the sub-problems to be similar enough to use the same HTTP status code. If they do not, the 207 (Multi-Status) [RFC4918] code could be used to encapsulate multiple status messages.

3.1. Problem Details Object Members

A problem details object can have the following members:

- "type" (string) - A URI reference [RFC3986] that identifies the problem type. When dereferenced, it is encouraged to provide human-readable documentation for the problem type (e.g., using HTML [W3C.REC-html5-20141028]). When this member is not present, its value is assumed to be "about:blank".

- "title" (string) - A short, human-readable summary of the problem type. It SHOULD NOT change from occurrence to occurrence of the problem, except for purposes of localisation (e.g., using proactive content negotiation; see [RFC7231], Section 3.4).

- "status" (number) - The HTTP status code ([RFC7231], Section 6) generated by the origin server for this occurrence of the problem.

- "detail" (string) - An human readable explanation specific to this occurrence of the problem.

- "instance" (string) - A URI reference that identifies the specific occurrence of the problem. It may or may not yield further information if dereferenced.

Consumers MUST use the type string as the primary identifier for the problem type; the title string is advisory, and included only for users who are not aware of the semantics of the URI, and don't have the ability to discover them (e.g., offline log analysis). Consumers SHOULD NOT automatically dereference the type URI.

The status member, if present, is only advisory; it conveys the HTTP status code used for the convenience of the consumer. Generators MUST use the same status code in the actual HTTP response, to assure that generic HTTP software that does not understand this format still behaves correctly. See Section 5 for further caveats regarding its use.

Consumers can use the status member to determine what the original status code used by the generator was, in cases where it has been changed (e.g., by an intermediary or cache), and when message bodies are persisted without HTTP information. Generic HTTP software will still use the HTTP status code.

The detail member, if present, ought to focus on helping the client correct the problem, rather than giving debugging information.

Consumers SHOULD NOT parse the detail member for information; extensions are more suitable and less error-prone ways to obtain such information.

Note that both "type" and "instance" accept relative URIs; this means that they must be resolved relative to the document's base URI, as per [RFC3986], Section 5.

3.2. Extension Members

Problem type definitions MAY extend the problem details object with additional members.

For example, our "out of credit" problem above defines two such extensions, "balance" and "accounts" to convey additional, problem-specific information.

Clients consuming problem details MUST ignore any such extensions that they don't recognise; this allows problem types to evolve and include additional information in the future.

Note that because extensions are effectively name spaced by the problem type, it is not possible to define new "standard" members without defining a new media type.

4. Defining New Problem Types

When an HTTP API needs to define a response that indicates an error condition, it might be appropriate to do so by defining a new problem type.

Before doing so, it's important to understand what they are good for, and what's better left to other mechanisms.

Problem details are not a debugging tool for the underlying implementation; rather, they are a way to expose greater detail about the HTTP interface itself. Designers of new problem types need to carefully consider the Security Considerations (Section 5); in particular the risk of exposing attack vectors by exposing implementation internals through error messages.

Likewise, truly generic problems - i.e., conditions that could potentially apply to any resource on the Web - are usually better expressed as plain status codes. For example, a "write access disallowed" problem is probably unnecessary, since a 403 Forbidden status code in response to a PUT request is self-explanatory.

Finally, an application might have a more appropriate way to carry an error in a format that it already defines. Problem details are intended to avoid the necessity of establishing new "fault" or "error" document formats, not to replace existing domain-specific formats.

That said, it is possible to add support for problem details to existing HTTP APIs using HTTP content negotiation (e.g., using the Accept request header to indicate a preference for this format; see [RFC7231], Section 5.3.2).

New problem type definitions MUST document:

1. A type URI (typically, with the "http" or "https" scheme),

2. A title that appropriately describes it (think short), and

3. The HTTP status code for it to be used with.

Problem type definitions MAY specify the use of the Retry-After response header ([RFC7231], Section 7.1.3) in appropriate circumstances.

A problem's type URI SHOULD resolve to HTML [W3C.REC-html5-20141028] documentation that explains how to resolve the problem.

A problem type definition MAY specify additional members on the Problem Details object. For example, an extension might use typed links [RFC5988] to another resource that can be used by machines to resolve the problem.

If such additional members are defined, their names SHOULD start with a letter (ALPHA, as per [RFC5234], Appendix B.1) and SHOULD consist of characters from ALPHA, DIGIT (Ibid.), and "_" (so that it can be serialized in formats other than JSON), and SHOULD be three characters or longer.

4.1. Example

For example, if you are publishing an HTTP API to your online shopping cart, you might need to indicate that the user is out of credit (our example from above), and therefore cannot make the purchase.

If you already have an application-specific format that can accommodate this information, it's probably best to do that. However, if you don't, you might consider using one of the problem details formats; JSON if your API is JSON-based, or XML if it uses that format.

To do so, you might look for an already-defined type URI that suits your purposes. If one is available, you can reuse that URI.

If one isn't available, you could mint and document a new type URI (which ought to be under your control and stable over time), an appropriate title and the HTTP status code that it will be used with, along with what it means and how it should be handled.

In summary: an instance URI will always identify a specific occurrence of a problem. On the other hand, type URIs can be reused if an appropriate description of a problem type is already available someplace else, or they can be created for new problem types.

4.2. Pre-Defined Problem Types

This specification reserves the use of one URI as a problem type:

The "about:blank" URI [RFC6694], when used as a problem type, indicates that the problem has no additional semantics beyond that of the HTTP status code.

When "about:blank" is used, the title SHOULD be the same as the recommended HTTP status phrase for that code (e.g., "Not Found" for

404, and so on), although it MAY be localized to suit client preferences (expressed with the Accept-Language request header).

Please note that according to how the "type" member is defined (Section 3.1), the "about:blank" URI is the default value for that member. Consequently, any problem details object not carrying an explicit "type" member implicitly uses this URI.

5. Security Considerations

When defining a new problem type, the information included must be carefully vetted. Likewise, when actually generating a problem - however it is serialized - the details given must also be scrutinized.

Risks include leaking information that can be exploited to compromise the system, access to the system, or the privacy of users of the system.

Generators providing links to occurrence information are encouraged to avoid making implementation details such as a stack dump available through the HTTP interface, since this can expose sensitive details of the server implementation, its data, and so on.

The "status" member duplicates the information available in the HTTP status code itself, thereby bringing the possibility of disagreement between the two. Their relative precedence is not clear, since a disagreement might indicate that (for example) an intermediary has modified the HTTP status code in transit (e.g., by a proxy or cache).

As such, those defining problem types as well as generators and consumers of problems need to be aware that generic software (such as proxies, load balancers, firewalls, virus scanners) are unlikely to know of or respect the status code conveyed in this member.

6. IANA Considerations

This specification defines two new Internet media types [RFC6838]:

Type name: application

Subtype name: problem+json

Required parameters: None

Optional parameters: None; unrecognised parameters should be ignored

Encoding considerations: Same as [RFC7159]

Security considerations: see Section 5 of this document

Interoperability considerations: None

Published specification: [this document]

Applications that use this media type: HTTP

Fragment identifier considerations: Same as for application/json ([RFC7159])

Additional information:

Deprecated alias names for this type: n/a

Magic number(s): n/a

File extension(s): n/a

Macintosh file type code(s): n/a

Person and email address to contact for further information: Mark Nottingham <mnot@mnot.net>

Intended usage: COMMON

Restrictions on usage: None.

Author: Mark Nottingham <mnot@mnot.net>

Change controller: IESG

Type name: application

Subtype name: problem+xml

Required parameters: None

Optional parameters: None; unrecognised parameters should be ignored

Encoding considerations: Same as [RFC7303]

Security considerations: see Section 5 of this document

Interoperability considerations: None

Published specification: [this document]

Applications that use this media type: HTTP

Fragment identifier considerations: Same as for application/xml (as specified by Section 5 of [RFC7303])

Additional information:

Deprecated alias names for this type: n/a

Magic number(s): n/a

File extension(s): n/a

Macintosh file type code(s): n/a

Person and email address to contact for further information: Mark No ttingham <mnot@mnot.net>

Intended usage: COMMON

Restrictions on usage: None.

Author: Mark Nottingham <mnot@mnot.net>

Change controller: IESG

7. Acknowledgements

The authors would like to thank Jan Algermissen, Mike Amundsen, Subbu Allamaraju, Roy Fielding, Eran Hammer, Sam Johnston, Mike McCall, Julian Reschke, and James Snell for review of this specification.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[W3C.REC-xml-20081126] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <http://www.w3.org/TR/2008/REC-xml-20081126>.

8.2. Informative References

[ISO-19757-2] International Organization for Standardization, "Information Technology --- Document Schema Definition Languages (DSDL) --- Part 2: Grammar-based Validation --- RELAX NG", ISO/IEC 19757-2, 2003.

[RFC4918] Dusseault, L., Ed., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, DOI 10.17487/RFC4918, June 2007, <http://www.rfc-editor.org/info/rfc4918>.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <http://www.rfc-editor.org/info/rfc5988>.

[RFC6694] Moonesamy, S., Ed., "The "about" URI Scheme", RFC 6694, DOI 10.17487/RFC6694, August 2012, <http://www.rfc-editor.org/info/rfc6694>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <http://www.rfc-editor.org/info/rfc6838>.

[RFC7303] Thompson, H. and C. Lilley, "XML Media Types", RFC 7303, DOI 10.17487/RFC7303, July 2014, <http://www.rfc-editor.org/info/rfc7303>.

[W3C.REC-html5-20141028] Hickson, I., Berjon, R., Faulkner, S., Leithead, T., Navara, E., O'Connor, E., and S. Pfeiffer, "HTML5", World Wide Web Consortium Recommendation REC-html5-20141028, October 2014, <http://www.w3.org/TR/2014/REC-html5-20141028>.

[W3C.REC-rdfa-core-20130822] Adida, B., Birbeck, M., McCarron, S., and I. Herman, "RDFa Core 1.1 - Second Edition", World Wide Web Consortium Recommendation REC-rdfa-core-20130822, August 2013, <http://www.w3.org/TR/2013/REC-rdfa-core-20130822>.

[W3C.REC-xml-stylesheet-20101028] Clark, J., Pieters, S., and H. Thompson, "Associating Style Sheets with XML documents 1.0 (Second Edition)", World Wide Web Consortium Recommendation REC-xml-stylesheet-20101028, October 2010, <http://www.w3.org/TR/2010/REC-xml-stylesheet-20101028>.

Appendix A. HTTP Problems and XML

Some HTTP-based APIs use XML [W3C.REC-xml-20081126] as their primary format convention. Such APIs can express problem details using the format defined in this appendix.

The RELAX NG schema [ISO-19757-2] for the XML format is as follows. Keep in mind that this schema is only meant as documentation, and not as a normative schema that captures all constraints of the XML format. Also, it would be possible to use other XML schema languages to define a similar set of constraints (depending on the features of the chosen schema language).

```
default namespace ns = "urn:ietf:rfc:XXXX"

start = problem

problem =
  element problem {
    (  element  type            { xsd:anyURI }?
     & element  title           { xsd:string }?
     & element  detail          { xsd:string }?
     & element  status          { xsd:positiveInteger }?
     & element  instance        { xsd:anyURI }? ),
    anyNsElement
  }

anyNsElement =
  (  element    ns:*  { anyNsElement | text }
   | attribute  *     { text })*
```

The media type for this format is "application/problem+xml".

Extension arrays and objects are serialized into the XML format by considering an element containing a child or children to represent an object, except for elements that contain only child element(s) named 'i', which are considered arrays. For example, the XML version of the example above appears in XML as follows:

```
HTTP/1.1 403 Forbidden
Content-Type: application/problem+xml
Content-Language: en

<?xml version="1.0" encoding="UTF-8"?>
<problem xmlns="urn:ietf:rfc:XXXX">
  <type>https://example.com/probs/out-of-credit</type>
  <title>You do not have enough credit.</title>
  <detail>Your current balance is 30, but that costs 50.</detail>
  <instance>https://example.net/account/12345/msgs/abc</instance>
  <balance>30</balance>
  <accounts>
    <i>https://example.net/account/12345</i>
    <i>https://example.net/account/67890</i>
  </accounts>
</problem>
```

Note that this format uses an XML Namespace. This is primarily to allow embedding it into other XML-based formats; it does not imply that it can or should be extended with elements or attributes in other namespaces. The RELAX NG schema explicitly only allows elements from the one namespace used in the XML format. Any

extension arrays and objects MUST be serialized into XML markup using only that namespace.

When using the XML format, it is possible to embed an XML Processing Instruction in the XML that instructs clients to transform the XML, using the referenced XSLT code [W3C.REC-xml-stylesheet-20101028]. If this code is transforming the XML into (X)HTML, then it is possible to serve the XML format, and yet have clients capable of performing the transformation display human-friendly (X)HTML that is rendered and displayed at the client. Note that when using this method, it is advisable to use XSLT 1.0, in order to maximize the number of clients capable of executing the XSLT code.

Appendix B. Using Problem Details with Other Formats

In some situations, it can be advantageous to embed Problem Details in formats other than those described here. For example, an API that uses HTML [W3C.REC-html5-20141028] might want to also use HTML for expressing its problem details.

Problem details can be embedded in other formats by either encapsulating one of the existing serializations (JSON or XML) into that format, or by translating the model of a Problem Detail (as specified in Section 3) into the format's conventions.

For example, in HTML, a problem could be embedded by encapsulating JSON in a script tag:

```
<script type="application/problem+json">
  {
   "type": "https://example.com/probs/out-of-credit",
   "title": "You do not have enough credit.",
   "detail": "Your current balance is 30, but that costs 50.",
   "instance": "/account/12345/msgs/abc",
   "balance": 30,
   "accounts": ["/account/12345",
                "/account/67890"]
  }
</script>
```

or by inventing a mapping into RDFa [W3C.REC-rdfa-core-20130822].

This specification does not make specific recommendations regarding embedding Problem Details in other formats; the appropriate way to embed them depends both upon the format in use and application of that format.

Authors' Addresses

Mark Nottingham
Akamai

Email: mnot@mnot.net
URI: http://www.mnot.net/

Erik Wilde

Email: erik.wilde@dret.net
URI: http://dret.net/netdret/

Network Working Group
Internet-Draft
Obsoletes: 3798 (if approved)
Updates: 2046, 3461 (if approved)
Intended status: Standards Track
Expires: June 4, 2017

T. Hansen, Ed.
AT&T Laboratories
A. Melnikov, Ed.
Isode Ltd

December 1, 2016

Message Disposition Notification
draft-ietf-appsawg-mdn-3798bis-16.txt

Abstract

This memo defines a MIME content-type that may be used by a mail user agent (MUA) or electronic mail gateway to report the disposition of a message after it has been successfully delivered to a recipient. This content-type is intended to be machine-processable. Additional message header fields are also defined to permit Message Disposition Notifications (MDNs) to be requested by the sender of a message. The purpose is to extend Internet Mail to support functionality often found in other messaging systems, such as X.400 and the proprietary "LAN-based" systems, and often referred to as "read receipts," "acknowledgements", or "receipt notifications." The intention is to do this while respecting privacy concerns, which have often been expressed when such functions have been discussed in the past.

Because many messages are sent between the Internet and other messaging systems (such as X.400 or the proprietary "LAN-based" systems), the MDN protocol is designed to be useful in a multi-protocol messaging environment. To this end, the protocol described in this memo provides for the carriage of "foreign" addresses, in addition to those normally used in Internet Mail. Additional attributes may also be defined to support "tunneling" of foreign notifications through Internet Mail.

This document obsoletes RFC 3798, moving it to Internet Standard. It also updates RFC 2046 (message/partial Media Type handling) and RFC 3461 (Original-Recipient header field generation requirement).

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/.

Copyright Notice

Table of Contents

1. Introduction

This memo defines a media type [RFC2046] for message disposition notifications (MDNs). An MDN can be used to notify the sender of a message of any of several conditions that may occur after successful delivery, such as display of the message contents, printing of the message, deletion (without display) of the message, or the recipient's refusal to provide MDNs. The "message/disposition-notification" content-type defined herein is intended for use within the framework of the "multipart/report" content type defined in RFC-REPORT [RFC6522].

This memo defines the format of the notifications and the RFC-MSGFMT [RFC5322] header fields used to request them.

This memo is an update to RFC 3798 and is intended to be published at Internet Standard Level.

1.1. Purposes

The MDNs defined in this memo are expected to serve several purposes:

a. Inform human beings of the disposition of messages after successful delivery, in a manner that is largely independent of human language;

b. Allow mail user agents to keep track of the disposition of messages sent, by associating returned MDNs with earlier message transmissions;

c. Convey disposition notification requests and disposition notifications between Internet Mail and "foreign" mail systems via a gateway;

d. Allow "foreign" notifications to be tunneled through a MIME-capable message system and back into the original messaging system that issued the original notification, or even to a third messaging system;

e. Allow language-independent, yet reasonably precise, indications of the disposition of a message to be delivered.

1.2. Requirements

These purposes place the following constraints on the notification protocol:

a. It must be readable by humans, and must be machine-parsable.

b. It must provide enough information to allow message senders (or their user agents) to unambiguously associate an MDN with the message that was sent and the original recipient address for which the MDN was issued (if such information is available), even if the message was forwarded to another recipient address.

c. It must also be able to describe the disposition of a message independent of any particular human language or of the terminology of any particular mail system.

d. The specification must be extensible in order to accommodate future requirements.

1.3. 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-KEYWORDS [RFC2119].

All syntax descriptions use the ABNF specified by RFC-MSGFMT [RFC5322], in which the lexical tokens (used below) are defined: "CRLF", "FWS", "CFWS", "field-name", "mailbox-list", "msg-id", and

"text". The following lexical tokens are defined in RFC-SMTP [RFC5321]: "atom".

2. Requesting Message Disposition Notifications

Message disposition notifications are requested by including a Disposition-Notification-To header field in the message containing one or more addresses specifying where dispositions should be sent. Further information to be used by the recipient's Mail User Agent (MUA) [RFC5598] in generating the MDN may be provided by also including Original-Recipient and/or Disposition-Notification-Options header fields in the message.

2.1. The Disposition-Notification-To Header

A request for the receiving user agent to issue message disposition notifications is made by placing a Disposition-Notification-To header field into the message. The syntax of the header field is

```
mdn-request-header = "Disposition-Notification-To" ":" mailbox-list CRLF
```

A Disposition-Notification-To header field can appear at most once in a message.

The presence of a Disposition-Notification-To header field in a message is merely a request for an MDN. The recipients' user agents are always free to silently ignore such a request.

An MDN MUST NOT itself have a Disposition-Notification-To header field. An MDN MUST NOT be generated in response to an MDN.

A user agent MUST NOT issue more than one MDN on behalf of each particular recipient. That is, once an MDN has been issued on behalf of a recipient, no further MDNs may be issued on behalf of that recipient by the same user agent, even if another disposition is performed on the message. However, if a message is forwarded, an MDN may have been issued for the recipient doing the forwarding and the recipient of the forwarded message may also cause an MDN to be generated.

It is also possible that if the same message is being accessed by multiple user agents (for example using POP3), then multiple dispositions might be generated for the same recipient. User agents SHOULD leverage support in the underlying message access protocol to prevent multiple MDNs from being generated. In particular, when the user agent is accessing the message using RFC-IMAP [RFC3501], it SHOULD implement the procedures specified in RFC-IMAP-MDN [RFC3503].

While Internet standards normally do not specify the behavior of user interfaces, it is strongly recommended that the user agent obtain the user's consent before sending an MDN. This consent could be obtained for each message through some sort of prompt or dialog box, or globally through the user's setting of a preference. The user might also indicate globally that MDNs are to never be sent. The purpose of obtaining user's consent is to protect user's privacy. The default value should be not to send MDNs.

MDNs MUST NOT be sent automatically if the address in the Disposition-Notification-To header field differs from the address in the Return-Path header field (see RFC-MSGFMT [RFC5322]). In this case, confirmation from the user MUST be obtained, if possible. If obtaining consent is not possible (e.g., because the user is not online at the time or the client is not an interactive email client), then an MDN MUST NOT be sent.

Confirmation from the user MUST be obtained (or no MDN sent) if there is no Return-Path header field in the message, or if there is more than one distinct address in the Disposition-Notification-To header field.

The comparison of the addresses is done using only the addr-spec (local-part "@" domain) portion, excluding any angle brackets, phrase and route. As prescribed by RFC 5322, the comparison is case-sensitive for the local-part and case-insensitive for the domain part. The local-part comparison SHOULD be done after performing local-part canonicalization (i.e. after removing the surrounding double-quote characters, if any, as well as any escaping "\" characters. (See RFC-MSGFMT [RFC5322] for more details.) Implementations MAY treat known domain aliases as equivalent for the purpose of comparison.

Note that use of subaddressing (see [RFC5233]) can result in a failure to match two local-parts and thus result in possible suppression of the MDN. This document doesn't recommend special handling for this case, as the receiving MUA can't reliably know whether or not the sender is using subaddressing.

If the message contains more than one Return-Path header field, the implementation may pick one to use for the comparison, or treat the situation as a failure of the comparison.

The reason for not automatically sending an MDN if the comparison fails or more than one address is specified is to reduce the possibility of mail loops and of MDNs being used for mail bombing.

It's especially important that a message that contains a Disposition-Notification-To header field also contain a Message-ID header field, to permit user agents to automatically correlate MDNs with their original messages.

If the request for message disposition notifications for some recipients and not others is desired, two copies of the message should be sent, one with a Disposition-Notification-To header field and one without. Many of the other header fields of the message (e.g., To, Cc) will be the same in both copies. The recipients in the respective message envelopes determine from whom message disposition notifications are requested and from whom they are not. If desired, the Message-ID header field may be the same in both copies of the message. Note that there are other situations (e.g., Bcc) in which it is necessary to send multiple copies of a message with slightly different header fields. The combination of such situations and the need to request MDNs for a subset of all recipients may result in more than two copies of a message being sent, some with a Disposition-Notification-To header field and some without.

If it is possible to determine that a recipient is a newsgroup, do not include a Disposition-Notification-To header field for that recipient. Similarly, if an existing message is resent or gatewayed to a newsgroup, the agent doing resending/gatewaying SHOULD strip the Disposition-Notification-To header field. See Section 5 for more discussion. Clients that see an otherwise valid Disposition-Notification-To header field in a newsgroup message SHOULD NOT generate an MDN.

2.2. The Disposition-Notification-Options Header

Extensions to this specification may require that information be supplied to the recipient's MUA for additional control over how and what MDNs are generated. The Disposition-Notification-Options header field provides an extensible mechanism for such information. The syntax of this header field is as follows:

```
Disposition-Notification-Options =
          "Disposition-Notification-Options" ":" [FWS]
                         disposition-notification-parameter-list CRLF

disposition-notification-parameter-list =
          disposition-notification-parameter
          *([FWS] ";" [FWS] disposition-notification-parameter)

disposition-notification-parameter = attribute [FWS] "="
          [FWS] importance [FWS] "," [FWS] value *([FWS] "," [FWS] value)

importance = "required" / "optional"

attribute = atom

value = word
```

A Disposition-Notification-Options header field can appear at most once in a message.

An importance of "required" indicates that interpretation of the disposition-notification-parameter is necessary for proper generation of an MDN in response to this request. An importance of "optional" indicates that an MUA that does not understand the meaning of this disposition-notification-parameter MAY generate an MDN in response anyway, ignoring the value of the disposition-notification-parameter.

No disposition-notification-parameter attribute names are defined in this specification. Attribute names may be defined in the future by later revisions or extensions to this specification. disposition-notification-parameter attribute names MUST be registered with the Internet Assigned Numbers Authority (IANA) using "Specification required" registration policy. The "X-" prefix has historically been used to denote unregistered "experimental" protocol elements, that are assumed not to become common use. Deployment experience of this and other protocols have shown that this assumption is often false. This document allows the use of the "X-" prefix primarily to allow the registration of attributes that are already in common use. The prefix has no meaning for new attributes. Its use in substantially new attributes may cause confusion and is therefore discouraged. (See Section 10 for a registration form.)

2.3. The Original-Recipient Header Field

Since electronic mail addresses may be rewritten while the message is in transit, it is useful for the original recipient address to be made available by the delivering Message Transfer Agent (MTA) [RFC5598]. The delivering MTA may be able to obtain this information

from the ORCPT parameter of the SMTP RCPT TO command, as defined in RFC-SMTP [RFC5321] and RFC-DSN-SMTP [RFC3461].

RFC-DSN-SMTP [RFC3461] is amended as follows: If the ORCPT information is available, the delivering MTA SHOULD insert an Original-Recipient header field at the beginning of the message (along with the Return-Path header field). The delivering MTA MAY delete any other Original-Recipient header fields that occur in the message. The syntax of this header field is as follows:

```
original-recipient-header =
          "Original-Recipient" ":" OWS address-type OWS ";" OWS generic-address
OWS

OWS      = [CFWS]
           ; Optional whitespace.
           ; MDN generators SHOULD use "*WSP"
           ; (typically a single space or nothing.
           ; It SHOULD be nothing at the end of a field),
           ; unless an RFC 5322 "comment" is required.
           ;
           ; MDN parsers MUST parse it as "[CFWS]".
```

The address-type and generic-address token are as specified in the description of the Original-Recipient field in Section 3.2.3.

The purpose of carrying the original recipient information and returning it in the MDN is to permit automatic correlation of MDNs with the original message on a per-recipient basis.

2.4. Use with the Message/Partial Media Type

The use of the header fields Disposition-Notification-To, Disposition-Notification-Options, and Original-Recipient with the MIME message/partial content type (RFC-MIME-MEDIA [RFC2046]]) requires further definition.

When a message is segmented into two or more message/partial fragments, the three header fields mentioned in the above paragraph SHOULD be placed in the "inner" or "enclosed" message (using the terms of RFC-MIME-MEDIA [RFC2046]). If these header fields are found in the header fields of any of the fragments, they are ignored.

When the multiple message/partial fragments are reassembled, the following applies. If these header fields occur along with the other header fields of a message/partial fragment message, they pertain to an MDN that will be generated for the fragment. If these header fields occur in the header fields of the "inner" or "enclosed" message (using the terms of RFC-MIME-MEDIA [RFC2046]), they pertain

to an MDN that will be generated for the reassembled message. Section 5.2.2.1 of RFC-MIME-MEDIA [RFC2046]) is amended to specify that, in addition to the header fields specified there, the three header fields described in this specification are to be appended, in order, to the header fields of the reassembled message. Any occurrences of the three header fields defined here in the header fields of the initial enclosing message MUST NOT be copied to the reassembled message.

3. Format of a Message Disposition Notification

A message disposition notification is a MIME message with a top-level content-type of multipart/report (defined in RFC-REPORT [RFC6522]). When multipart/report content is used to transmit an MDN:

a. The report-type parameter of the multipart/report content is "disposition-notification".

b. The first component of the multipart/report contains a human-readable explanation of the MDN, as described in RFC-REPORT [RFC6522].

c. The second component of the multipart/report is of content-type message/disposition-notification, described in Section 3.1 of this document.

d. If the original message or a portion of the message is to be returned to the sender, it appears as the third component of the multipart/report. The decision of whether or not to return the message or part of the message is up to the MUA generating the MDN. However, in the case of encrypted messages requesting MDNs, if the original message or a portion thereof is returned, it MUST be in its original encrypted form.

NOTE: For message disposition notifications gatewayed from foreign systems, the header fields of the original message may not be available. In this case, the third component of the MDN may be omitted, or it may contain "simulated" RFC-MSGFMT [RFC5322] header fields that contain equivalent information. In particular, it is very desirable to preserve the subject and date fields from the original message.

The MDN MUST be addressed (in both the message header field and the transport envelope) to the address(es) from the Disposition-

Notification-To header field from the original message for which the MDN is being generated.

The From header field of the MDN MUST contain the address of the person for whom the message disposition notification is being issued.

The envelope sender address (i.e., SMTP "MAIL FROM") of the MDN MUST be null (<>), specifying that no Delivery Status Notification messages nor other messages indicating successful or unsuccessful delivery are to be sent in response to an MDN.

A message disposition notification MUST NOT itself request an MDN. That is, it MUST NOT contain a Disposition-Notification-To header field.

The Message-ID header field (if present) for an MDN MUST be different from the Message-ID of the message for which the MDN is being issued.

A particular MDN describes the disposition of exactly one message for exactly one recipient. Multiple MDNs may be generated as a result of one message submission, one per recipient. However, due to the circumstances described in Section 2.1, it's possible that some of the recipients for whom MDNs were requested will not generate MDNs.

3.1. The message/disposition-notification Media Type

The message/disposition-notification Media Type is defined as follows:

Type name: message

Subtype name: disposition-notification

Required parameters: none

Optional parameters: none

Encoding considerations: "7bit" encoding is sufficient and MUST be used to maintain readability when viewed by non-MIME mail readers.

Security considerations: discussed in Section 6 of [RFCXXXX].

Interoperability considerations: none

Published specification: [RFCXXXX]

Applications that use this media type: Mail Transfer Agents and email clients that support multipart/report generation and/or parsing.

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A

Magic number(s): none

File extension(s): .disposition-notification

Macintosh file type code(s): The 'TEXT' type code is suggested as files of this type are typically used for diagnostic purposes and suitable for analysis in a text editor. A uniform type identifier (UTI) of "public.utf8-email-message-header" is suggested. This type conforms to "public.plain-text".

Person & email address to contact for further information: ART Area Mailing List <art@ietf.org>

Intended usage: COMMON

Restrictions on usage: This media type contains textual data in the US-ASCII charset, which is always 7-bit.

Author: See the Authors' Addresses section of [RFCXXXX]

Change controller: IETF

Provisional registration? no

(While the 7bit restriction applies to the message/disposition-notification portion of the multipart/report content, it does not apply to the optional third portion of the multipart/report content.)

The message/disposition-notification report type for use in the multipart/report is "disposition-notification".

The body of a message/disposition-notification consists of one or more "fields" formatted according to the ABNF of RFC-MSGFMT [RFC5322] header "fields". The syntax of the message/disposition-notification content is as follows:

```
disposition-notification-content = [ reporting-ua-field CRLF ]
          [ mdn-gateway-field CRLF ]
          [ original-recipient-field CRLF ]
          final-recipient-field CRLF
          [ original-message-id-field CRLF ]
          disposition-field CRLF
          *( error-field CRLF )
          *( extension-field CRLF )

extension-field = extension-field-name ":" *([FWS] text)

extension-field-name = field-name
```

Note that the order of the above fields is recommended, but not fixed. Extension fields can appear anywhere.

3.1.1. General conventions for fields

Since these fields are defined according to the rules of RFC-MSGFMT [RFC5322], the same conventions for continuation lines and comments apply. Notification fields may be continued onto multiple lines by beginning each additional line with a SPACE or HTAB. Text that appears in parentheses is considered a comment and not part of the contents of that notification field. Field names are case-insensitive, so the names of notification fields may be spelled in any combination of upper and lower case letters. [RFC5322] comments in notification fields may use the "encoded-word" construct defined in RFC-MIME-HEADER [RFC2047].

3.1.2. "*-type" subfields

Several fields consist of a "-type" subfield, followed by a semi-colon, followed by "*text".
For these fields, the keyword used in the address-type or MTA-type subfield indicates the expected format of the address or MTA-name that follows.

The "-type" subfields are defined as follows:

a. An "address-type" specifies the format of a mailbox address. For example, Internet Mail addresses use the "rfc822" address-type. Other values can appear in this field as specified in the "Address Types" IANA subregistry established by RFC-DSN-FORMAT [RFC3464].

```
address-type = atom

atom = <The version from RFC 5321 (not from RFC 5322) is used in this document.>
```

b. An "MTA-name-type" specifies the format of a mail transfer agent name. For example, for an SMTP server on an Internet host, the MTA name is the domain name of that host, and the "dns" MTA-name-type is used. Other values can appear in this field as specified in the "MTA Name Types" IANA subregistry established by RFC-DSN-FORMAT [RFC3464].

```
mta-name-type = atom
```

Values for address-type and mta-name-type are case-insensitive. Thus, address-type values of "RFC822" and "rfc822" are equivalent.

The Internet Assigned Numbers Authority (IANA) maintains a registry of address-type and mta-name-type values, along with descriptions of the meanings of each, or a reference to one or more specifications that provide such descriptions. (The "rfc822" address-type is defined in RFC-DSN-SMTP [RFC3461].) Registration forms for address-type and mta-name-type appear in RFC-DSN-FORMAT [RFC3464].

3.2. Message/disposition-notification Content Fields

3.2.1. The Reporting-UA field

```
reporting-ua-field = "Reporting-UA" ":" OWS ua-name OWS [ ";" OWS ua-product OWS
 ]

ua-name = *text-no-semi

ua-product = *([FWS] text)

text-no-semi = %d1-9 /          ; "text" characters excluding NUL, CR,
               %d11 / %d12 / %d14-58 / %d60-127  ; LF, or semi-colon
```

The Reporting-UA field is defined as follows:

An MDN describes the disposition of a message after it has been delivered to a recipient. In all cases, the Reporting-UA is the MUA that performed the disposition described in the MDN.

The "Reporting-UA" field contains information about the MUA that generated the MDN, which is often used by servers to help identify the scope of reported interoperability problems, to work around or tailor responses to avoid particular MUA limitations, and for analytics regarding MUA or operating system use. A MUA SHOULD send a "Reporting-UA" field unless specifically configured not to do so.

If the reporting MUA consists of more than one component (e.g., a base program and plug-ins), this may be indicated by including a list of product names.

A reporting MUA SHOULD limit generated product identifiers to what is necessary to identify the product; a sender MUST NOT generate advertising or other nonessential information within the product identifier.

A reporting MUA SHOULD NOT generate a "Reporting-UA" field containing needlessly fine-grained detail and SHOULD limit the addition of subproducts by third parties. Overly long and detailed "Reporting-UA" field values increase the risk of a user being identified against their wishes ("fingerprinting").

Likewise, implementations are encouraged not to use the product tokens of other implementations in order to declare compatibility with them, as this circumvents the purpose of the field. If a MUA masquerades as a different MUA, recipients can assume that the user intentionally desires to see responses tailored for that identified MUA, even if they might not work as well for the actual MUA being used.

Example:

Reporting-UA: Foomail 97.1

3.2.2. The MDN-Gateway field

The MDN-Gateway field indicates the name of the gateway or MTA that translated a foreign (non-Internet) message disposition notification into this MDN. This field MUST appear in any MDN that was translated by a gateway from a foreign system into MDN format, and MUST NOT appear otherwise.

```
mdn-gateway-field = "MDN-Gateway" ":" OWS mta-name-type OWS ";" OWS mta-name OWS

mta-name = *text
```

For gateways into Internet Mail, the MTA-name-type will normally be "dns", and the mta-name will be the Internet domain name of the gateway.

3.2.3. Original-Recipient field

The Original-Recipient field indicates the original recipient address as specified by the sender of the message for which the MDN is being issued. For Internet Mail messages, the value of the Original-Recipient field is obtained from the Original-Recipient header field from the message for which the MDN is being generated. If there is an Original-Recipient header field in the message, or if information about the original recipient is reliably available some other way, then the Original-Recipient field MUST be included. Otherwise, the Original-Recipient field MUST NOT be included. If there is more than one Original-Recipient header field in the message, the MUA may choose the one to use, or act as if no Original-Recipient header field is present.

```
original-recipient-field =
          "Original-Recipient" ":" OWS address-type OWS ";" OWS generic-address
OWS

generic-address = *text
```

The address-type field indicates the type of the original recipient address. If the message originated within the Internet, the address-type field will normally be "rfc822", and the address will be according to the syntax specified in RFC-MSGFMT [RFC5322]. The value "unknown" should be used if the Reporting MUA cannot determine the type of the original recipient address from the message envelope. This address is the same as that provided by the sender and can be used to automatically correlate MDN reports with original messages on a per recipient basis.

3.2.4. Final-Recipient field

The Final-Recipient field indicates the recipient for which the MDN is being issued. This field MUST be present.

The syntax of the field is as follows:

```
final-recipient-field =
          "Final-Recipient" ":" OWS address-type OWS ";" OWS generic-address OWS
```

The generic-address subfield of the Final-Recipient field SHOULD contain the mailbox address of the recipient (which will be the same as the From header field of the MDN) as it was when the MDN was generated by the MUA.

> One example of when this field might not contain the final recipient address of the message is when an alias (e.g. "customer-support@example.com") forwards mail to a specific personal address (e.g. "bob@example.com"). Bob might want to be able to send MDNs, but not give away his personal email address. In this case the Final-Recipient field can be "customer-support@example.com" instead of "bob@example.com".

The Final-Recipient address may differ from the address originally provided by the sender, because it may have been transformed during forwarding and gatewaying into a totally unrecognizable mess. However, in the absence of the optional Original-Recipient field, the Final-Recipient field and any returned content may be the only information available with which to correlate the MDN with a particular message recipient.

The address-type subfield indicates the type of address expected by the reporting MTA in that context. Recipient addresses obtained via SMTP will normally be of address-type "rfc822", but can be other values from the "Address Types" subregistry of the "Delivery Status Notification (DSN) Types" IANA registry.

Since mailbox addresses (including those used in the Internet) may be case sensitive, the case of alphabetic characters in the address MUST be preserved.

3.2.5. Original-Message-ID field

The Original-Message-ID field indicates the message-ID of the message for which the MDN is being issued. It is obtained from the Message-ID header field of the message for which the MDN is issued. This field MUST be present if and only if the original message contained a Message-ID header field. The syntax of the field is as follows:

```
original-message-id-field =
          "Original-Message-ID" ":" msg-id
```

The msg-id token is as specified in RFC-MSGFMT [RFC5322].

3.2.6. Disposition field

The Disposition field indicates the action performed by the Reporting-MUA on behalf of the user. This field MUST be present.

The syntax for the Disposition field is:

```
disposition-field =
          "Disposition" ":" OWS disposition-mode OWS ";"
          OWS disposition-type
          [ OWS "/" OWS disposition-modifier
          *( OWS "," OWS disposition-modifier ) ] OWS

disposition-mode = action-mode OWS "/" OWS sending-mode

action-mode = "manual-action" / "automatic-action"

sending-mode = "MDN-sent-manually" / "MDN-sent-automatically"

disposition-type = "displayed" / "deleted" / "dispatched" /
          "processed"

disposition-modifier = "error" / disposition-modifier-extension

disposition-modifier-extension = atom
```

The disposition-mode, disposition-type, and disposition-modifier values may be spelled in any combination of upper and lower case US-ASCII characters.

3.2.6.1. Disposition modes

Disposition mode consists of 2 parts: action mode and sending mode.

The following action modes are defined:

"manual-action" The disposition described by the disposition type was a result of an explicit instruction by the user rather than some sort of automatically performed action. (This might include the case when the user has manually configured her MUA to automatically respond to valid MDN requests.) Unless prescribed otherwise in a particular mail environment, in order to preserve user's privacy, this MUST be the default for MUAs.

"automatic-action" The disposition described by the disposition type was a result of an automatic action, rather than an explicit instruction by the user for this message. This is typically generated by a Mail Delivery Agent (e.g. MDN generations by Sieve reject action [RFC5429], Fax-over-Email [RFC3249], Voice Messaging System (VPIM) [RFC3801] or upon delivery to a mailing list).

"Manual-action" and "automatic-action" are mutually exclusive. One or the other MUST be specified.

The following sending modes are defined:

"MDN-sent-manually" The user explicitly gave permission for this particular MDN to be sent. Unless prescribed otherwise in a particular mail environment, in order to preserve user's privacy, this MUST be the default for MUAs.

"MDN-sent-automatically" The MDN was sent because the MUA had previously been configured to do so automatically.

"MDN-sent-manually" and "MDN-sent-automatically" are mutually exclusive. One or the other MUST be specified.

3.2.6.2. Disposition types

The following disposition-types are defined:

"displayed" The message has been displayed by the MUA to someone reading the recipient's mailbox. There is no guarantee that the content has been read or understood.

"dispatched" The message has been sent somewhere in some manner (e.g., printed, faxed, forwarded) without necessarily having been previously displayed to the user. The user may or may not see the message later.

"processed" The message has been processed in some manner (i.e., by some sort of rules or server) without being displayed to the user. The user may or may not see the message later, or there may not even be a human user associated with the mailbox.

"deleted" The message has been deleted. The recipient may or may not have seen the message. The recipient might "undelete" the message at a later time and read the message.

3.2.6.3. Disposition modifiers

Only the extension disposition modifiers is defined:

disposition-modifier-extension
Disposition modifiers may be defined in the future by later revisions or extensions to this specification. MDN disposition value names MUST be registered with the Internet Assigned Numbers Authority (IANA) using "Specification required" registration policy. (See Section 10 for a registration form.) MDNs with disposition modifier names not understood by the receiving MUA MAY be silently ignored or placed in the user's mailbox without special interpretation. They MUST NOT cause any error message to be sent to the sender of the MDN.

It is not required that an MUA be able to generate all of the possible values of the Disposition field.

A user agent MUST NOT issue more than one MDN on behalf of each particular recipient. That is, once an MDN has been issued on behalf of a recipient, no further MDNs may be issued on behalf of that recipient, even if another disposition is performed on the message. However, if a message is forwarded, a "dispatched" MDN MAY be issued for the recipient doing the forwarding and the recipient of the forwarded message may also cause an MDN to be generated.

3.2.7. Error Field

The Error field is used to supply additional information in the form of text messages when the "error" disposition modifier appear. The syntax is as follows:

error-field = "Error" ":" *([FWS] text)

Note that syntax of these header fields doesn't include comments, so "encoded-word" construct defined in RFC-MIME-HEADER [RFC2047] can't be used to convey non ASCII text. Application that need to convey non ASCII text in these fields should consider implementing message/global-disposition-notification media type specified in [RFC6533] instead of this specification.

3.3. Extension-fields

Additional MDN fields may be defined in the future by later revisions or extensions to this specification. MDN field names MUST be registered with the Internet Assigned Numbers Authority (IANA) using "Specification required" registration policy. (See Section 10 for a registration form.) MDN Extension-fields may be defined for the following reasons:

a. To allow additional information from foreign disposition reports to be tunneled through Internet MDNs. The names of such MDN fields should begin with an indication of the foreign environment name (e.g., X400-Physical-Forwarding-Address).

b. To allow transmission of diagnostic information that is specific to a particular mail user agent (MUA). The names of such MDN fields should begin with an indication of the MUA implementation that produced the MDN (e.g., Foomail-information).

4. Timeline of events

The following timeline shows when various events in the processing of a message and generation of MDNs take place:

-- User composes message

-- User tells MUA to send message.

-- MUA passes message to Mail Submission Agent (MSA), original recipient information passed along.

-- MSA sends message to next MTA.

-- Final MTA receives message.

-- Final MTA delivers message to recipient's mailbox (possibly generating a Delivery Status Notification (DSN)).

-- (Recipient's) MUA discovers a new message in recipient's mailbox and decides whether an MDN should be generated. If the MUA has information that an MDN has already been generated for this message, no further MDN processing described below is performed. If MUA decides that no MDN can be generated, no further MDN processing described below is performed.

-- MUA performs automatic processing and might generate corresponding MDNs ("dispatched", "processed" or "deleted" disposition type with "automatic-action" and "MDN-sent-automatically" disposition modes). The MUA remembers that an MDN was generated.

-- MUA displays list of messages to user.

-- User selects a message and requests that some action be performed on it.

-- MUA performs requested action; if an automatic MDN has not already been generated, with user's permission, sends an appropriate MDN ("displayed", "dispatched", "processed", or "deleted" disposition type, with "manual-action" and "MDN-sent-manually" or "MDN-sent-automatically" disposition mode). The MUA remembers that an MDN was generated.

-- User possibly performs other actions on message, but no further MDNs are generated.

5. Conformance and Usage Requirements

An MUA or gateway conforms to this specification if it generates MDNs according to the protocol defined in this memo. It is not necessary to be able to generate all of the possible values of the Disposition field.

MUAs and gateways MUST NOT generate the Original-Recipient field of an MDN unless the mail protocols provide the address originally

specified by the sender at the time of submission. Ordinary SMTP does not make that guarantee, but the SMTP extension defined in RFC-DSN-SMTP [RFC3461] permits such information to be carried in the envelope if it is available. The Original-Recipient header field defined in this document provides a way for the MTA to pass the original recipient address to the MUA.

Each sender-specified recipient address may result in more than one MDN. If an MDN is requested for a recipient that is forwarded to multiple recipients of an "alias" (as defined in RFC-DSN-SMTP [RFC3461], section 6.2.7.3), each of the recipients may issue an MDN.

Successful distribution of a message to a mailing list exploder or gateway to Usenet newsgroup SHOULD be considered the final disposition of the message. A mailing list exploder MAY issue an MDN with a disposition type of "processed" and disposition modes of "automatic-action" and "MDN-sent-automatically" indicating that the message has been forwarded to the list. In this case, the request for MDNs is not propagated to the members of the list.

Alternatively (if successful distribution of a message to a mailing list exploder/Usenet newsgroup is not considered the final disposition of the message), the mailing list exploder can issue no MDN and propagate the request for MDNs to all members of the list. The latter behavior is not recommended for any but small, closely knit lists, as it might cause large numbers of MDNs to be generated and may cause confidential subscribers to the list to be revealed. The mailing list exploder can also direct MDNs to itself, correlate them, and produce a report to the original sender of the message.

This specification places no restrictions on the processing of MDNs received by user agents or mailing lists.

6. Security Considerations

The following security considerations apply when using MDNs:

6.1. Forgery

MDNs can be (and are, in practice) forged as easily as ordinary Internet electronic mail. User agents and automatic mail handling facilities (such as mail distribution list exploders) that wish to make automatic use of MDNs should take appropriate precautions to minimize the potential damage from denial-of-service attacks.

Security threats related to forged MDNs include the sending of:

a. A falsified disposition notification when the indicated disposition of the message has not actually occurred,

b. Unsolicited MDNs

Similarly, a forged spam or phishing email message can contain Disposition-Notification-To header field that can trick the recipient to send an MDN. MDN processing should only be invoked once authenticity of email message is verified.

6.2. Privacy

Another dimension of security is privacy. There may be cases in which a message recipient does not wish the disposition of messages addressed to him to be known, or is concerned that the sending of MDNs may reveal other sensitive information (e.g., when the message was read, using which email client, which OS was used). In this situation, it is acceptable for the MUA to silently ignore requests for MDNs.

If the Disposition-Notification-To header field is passed on unmodified when a message is distributed to the subscribers of a mailing list, the subscribers to the list may be revealed to the sender of the original message by the generation of MDNs.

Headers of the original message returned in part 3 of the multipart/report, as well as content of the message/disposition-notification part could reveal confidential information about host names and/or network topology inside a firewall.

Disposition mode (Section 3.2.6.1) can leak information about recipient's MUA configuration, in particular whether MDNs are acknowledged manually or automatically. If this is a concern, MUAs can return "manual-action/MDN-sent-manually" disposition mode in generated MDNs.

In general, any optional MDN field may be omitted if the Reporting MUA site or user determines that inclusion of the field would impose too great a compromise of site confidentiality. The need for such confidentiality must be balanced against the utility of the omitted information in MDNs.

In some cases, someone with access to the message stream may use the MDN request mechanism to monitor the mail reading habits of a target. If the target is known to generate MDN reports, they could add a disposition-notification-to field containing the envelope from

address. This risk can be minimized by not sending MDN's automatically.

6.2.1. Disclosure of Product Information

The "Reporting-UA" field (Section 3.2.1) User-Agent (Section 5.5.3) and header fields often reveal information about the respective sender's software systems. In theory, this can make it easier for an attacker to exploit known security holes; in practice, attackers tend to try all potential holes regardless of the apparent software versions being used. Also note that the "Reporting-UA" field doesn't provide any new information in comparison to the "User-Agent" and/or (undocumented) "X-Mailer" header fields used by many MUAs.

6.2.2. MUA Fingerprinting

The "Reporting-UA" field (Section 3.2.1) might contain enough information to uniquely identify a specific device, usually when combined with other characteristics, particularly if the user agent sends excessive details about the user's system or extensions. Even when the guidance in Section 3.2.1 is followed to avoid fingerprinting, other sources of unique information may still be present, such as the Accept-Language header fields.

6.3. Non-Repudiation

MDNs do not provide non-repudiation with proof of delivery. Within the framework of today's Internet Mail, the MDNs defined in this document provide valuable information to the mail user; however, MDNs cannot be relied upon as a guarantee that a message was or was not seen by the recipient. Even if MDNs are not actively forged, they may be lost in transit. The recipient may bypass the MDN issuing mechanism in some manner.

One possible solution for this purpose can be found in RFC-SEC-SERVICES [RFC2634].

6.4. Mail Bombing

The MDN request mechanism introduces an additional way of mailbombing a mailbox. The MDN request notification provides an address to which MDN's should be sent. It is possible for an attacking agent to send a potentially large set of messages to otherwise unsuspecting third party recipients with a false "disposition-notification-to:" address. Automatic, or simplistic processing of such requests would result in a flood of MDN notifications to the target of the attack. Additionally, as generated MDN notifications can include full content of messages that caused them and thus they can be bigger than such

messages, they can be used for bandwidth amplification attacks. Such an attack could overrun the storage capacity of the targeted mailbox and/or of the mail transport system, and deny service.

For that reason, MDN's SHOULD NOT be sent automatically where the "disposition-notification-to:" address is different from the SMTP "MAIL FROM" address (which is carried in the Return-Path header field). See Section 2.1 for further discussion.

7. Collected ABNF Grammar

NOTE: The following lexical tokens are defined in RFC-MSGFMT [RFC5322]: CRLF, FWS, CFWS, field-name, mailbox-list, msg-id, text, comment, word. The following lexical tokens are defined in RFC-SMTP [RFC5321]: atom. (Note that RFC-MSGFMT [RFC5322] also defines "atom", but the version from RFC-SMTP [RFC5321] is more restrictive and this more restrictive version is used in this document.) "encoded-word" construct defined in RFC-MIME-HEADER [RFC2047] is allowed everywhere where RFC-MSGFMT [RFC5322] "comment" is used, for example in CFWS.

```
OWS          = [CFWS]
                 ; Optional whitespace.
                 ; MDN generators SHOULD use "*WSP"
                 ; (typically a single space or nothing.
                 ; It SHOULD be nothing at the end of a field),
                 ; unless an RFC 5322 "comment" is required.
                 ;
                 ; MDN parsers MUST parse it as "[CFWS]".

Message header fields:
   mdn-request-header =
          "Disposition-Notification-To" ":" mailbox-list CRLF

   Disposition-Notification-Options =
          "Disposition-Notification-Options" ":" [FWS]
                    disposition-notification-parameter-list CRLF

   disposition-notification-parameter-list =
                    disposition-notification-parameter
                    *([FWS] ";" [FWS] disposition-notification-parameter)

   disposition-notification-parameter = attribute [FWS] "=" [FWS]
                    importance [FWS] "," [FWS] value *([FWS] "," [FWS] value)

   importance = "required" / "optional"

   attribute = atom
```

```
   value = word

   original-recipient-header =
          "Original-Recipient" ":" OWS address-type OWS ";" OWS generic-address
OWS CRLF

Report content:
   disposition-notification-content =
          [ reporting-ua-field CRLF ]
          [ mdn-gateway-field CRLF ]
          [ original-recipient-field CRLF ]
          final-recipient-field CRLF
          [ original-message-id-field CRLF ]
          disposition-field CRLF
          *( error-field CRLF )
          *( extension-field CRLF )

   address-type = atom

   mta-name-type = atom

   reporting-ua-field = "Reporting-UA" ":" OWS ua-name OWS [ ";" OWS ua-product
OWS ]

   ua-name = *text-no-semi

   ua-product = *([FWS] text)

   text-no-semi = %d1-9 /        ; "text" characters excluding NUL, CR,
           %d11 / %d12 / %d14-58 / %d60-127      ; LF, or semi-colon

   mdn-gateway-field = "MDN-Gateway" ":" OWS mta-name-type OWS ";" OWS mta-name

   mta-name = *text

   original-recipient-field =
          "Original-Recipient" ":" OWS address-type OWS ";" OWS generic-address
OWS

   generic-address = *text

   final-recipient-field =
          "Final-Recipient" ":" OWS address-type OWS ";" OWS generic-address OWS

   original-message-id-field = "Original-Message-ID" ":" msg-id

   disposition-field =
          "Disposition" ":" OWS disposition-mode OWS ";"
          OWS disposition-type
          [ OWS "/" OWS disposition-modifier
          *( OWS "," OWS disposition-modifier ) ] OWS
```

```
disposition-mode = action-mode OWS "/" OWS sending-mode

action-mode = "manual-action" / "automatic-action"

sending-mode = "MDN-sent-manually" / "MDN-sent-automatically"

disposition-type = "displayed" / "deleted" / "dispatched" /
        "processed"

disposition-modifier = "error" / disposition-modifier-extension

disposition-modifier-extension = atom

error-field = "Error" ":" *([FWS] text)

extension-field = extension-field-name ":" *([FWS] text)

extension-field-name = field-name
```

8. Guidelines for Gatewaying MDNs

NOTE: This section provides non-binding recommendations for the construction of mail gateways that wish to provide semi-transparent disposition notifications between the Internet and another electronic mail system. Specific MDN gateway requirements for a particular pair of mail systems may be defined by other documents.

8.1. Gatewaying from other mail systems to MDNs

A mail gateway may issue an MDN to convey the contents of a "foreign" disposition notification over Internet Mail. When there are appropriate mappings from the foreign notification elements to MDN fields, the information may be transmitted in those MDN fields. Additional information (such as might be needed to tunnel the foreign notification through the Internet) may be defined in extension MDN fields. (Such fields should be given names that identify the foreign mail protocol, e.g., X400-* for X.400 protocol elements).

The gateway must attempt to supply reasonable values for the Reporting-UA, Final-Recipient, and Disposition fields. These will normally be obtained by translating the values from the foreign notification into their Internet-style equivalents. However, some loss of information is to be expected.

The sender-specified recipient address and the original message-id, if present in the foreign notification, should be preserved in the Original-Recipient and Original-Message-ID fields.

The gateway should also attempt to preserve the "final" recipient address from the foreign system. Whenever possible, foreign protocol elements should be encoded as meaningful printable ASCII strings.

For MDNs produced from foreign disposition notifications, the name of the gateway MUST appear in the MDN-Gateway field of the MDN.

8.2. Gatewaying from MDNs to other mail systems

It may be possible to gateway MDNs from the Internet into a foreign mail system. The primary purpose of such gatewaying is to convey disposition information in a form that is usable by the destination system. A secondary purpose is to allow "tunneling" of MDNs through foreign mail systems in case the MDN may be gatewayed back into the Internet.

In general, the recipient of the MDN (i.e., the sender of the original message) will want to know, for each recipient: the closest available approximation to the original recipient address, and the disposition (displayed, printed, etc.).

If possible, the gateway should attempt to preserve the Original-Recipient address and Original-Message-ID (if present) in the resulting foreign disposition report.

If it is possible to tunnel an MDN through the destination environment, the gateway specification may define a means of preserving the MDN information in the disposition reports used by that environment.

8.3. Gatewaying of MDN-requests to other mail systems

By use of the separate disposition-notification-to request header field, this specification offers a richer functionality than most, if not all, other email systems. In most other email systems, the notification recipient is identical to the message sender as indicated in the "from" address. There are two interesting cases when gatewaying into such systems:

1. If the address in the disposition-notification-to header field is identical to the address in the SMTP "MAIL FROM", the expected behavior will result, even if the disposition-notification-to information is lost. Systems should propagate the MDN request.

2. If the address in the disposition-notification-to header field is different from the address in the SMTP "MAIL FROM", gatewaying

into a foreign system without a separate notification address will result in unintended behavior. This is especially important when the message arrives via a mailing list expansion software that may specifically replace the SMTP "MAIL FROM" address with an alternate address. In such cases, the MDN request should not be gatewayed and should be silently dropped. This is consistent with other forms of non-support for MDN.

9. Example

NOTE: This example is provided as illustration only, and is not considered part of the MDN protocol specification. If the example conflicts with the protocol definition above, the example is wrong.

Likewise, the use of *-type subfield names or extension fields in this example is not to be construed as a definition for those type names or extension fields.

This is an MDN issued after a message has been displayed to the user of an Internet Mail user agent.

```
Date: Wed, 20 Sep 1995 00:19:00 (EDT) -0400
From: Joe Recipient <Joe_Recipient@example.com>
Message-Id: <199509200019.12345@example.com>
Subject: Disposition notification
To: Jane Sender <Jane_Sender@example.org>
MIME-Version: 1.0
Content-Type: multipart/report; report-type=disposition-notification;
   boundary="RAA14128.773615765/example.com"

--RAA14128.773615765/example.com

The message sent on 1995 Sep 19 at 13:30:00 (EDT) -0400 to Joe
Recipient <Joe_Recipient@example.com> with subject "First draft of
report" has been displayed.
This is no guarantee that the message has been read or understood.

--RAA14128.773615765/example.com
content-type: message/disposition-notification

Reporting-UA: joes-pc.cs.example.com; Foomail 97.1
Original-Recipient: rfc822;Joe_Recipient@example.com
Final-Recipient: rfc822;Joe_Recipient@example.com
Original-Message-ID: <199509192301.23456@example.org>
Disposition: manual-action/MDN-sent-manually; displayed

--RAA14128.773615765/example.com
content-type: message/rfc822

[original message optionally goes here]

--RAA14128.773615765/example.com--
```

10. IANA Considerations

There are two actions for IANA:

1. IANA is asked to update the registration template for the message/disposition-notification media type to the one in Section 3.1 of this document, and to update the reference for that media type to point to this document instead of to RFC 3798.

2. The registries specified here already exist, and this section is updating their documentation. IANA is asked to change the reference document for the three Message Disposition Notification Parameters registries to point to this document instead of to RFC 3798.

This document specifies three types of parameters that must be registered with the Internet Assigned Numbers Authority (IANA). All of them use [RFC5226] "Specification required" IANA registration policy.

The forms below are for use when registering a new disposition-notification-parameter name for the Disposition-Notification-Options header field, a new disposition modifier name, or a new MDN extension field. Each piece of information required by a registration form may be satisfied either by providing the information on the form itself, or by including a reference to a published, publicly available specification that includes the necessary information. IANA MAY reject registrations because of incomplete registration forms or incomplete specifications.

To register, complete the following applicable form and send it via electronic mail to <IANA@IANA.ORG>.

10.1. Disposition-Notification-Options header field disposition-notification-parameter names

A registration for a Disposition-Notification-Options header field disposition-notification-parameter name MUST include the following information:

a. The proposed disposition-notification-parameter name.

b. The syntax for disposition-notification-parameter values, specified using BNF, ABNF, regular expressions, or other non-ambiguous language.

c. If disposition-notification-parameter values are not composed entirely of graphic characters from the US-ASCII repertoire, a specification for how they are to be encoded as graphic US-ASCII characters in a Disposition-Notification-Options header field.

d. A reference to a permanent and readily available public specification that describes the semantics of the disposition-notification-parameter values.

10.2. Disposition modifier names

A registration for a disposition-modifier name (used in the Disposition field of a message/disposition-notification) MUST include the following information:

a. The proposed disposition-modifier name.

b. A reference to a permanent and readily available public specification that describes the semantics of the disposition modifier.

10.3. MDN extension field names

A registration for an MDN extension-field name MUST include the following information:

a. The proposed extension field name.

b. The syntax for extension values, specified using BNF, ABNF, regular expressions, or other non-ambiguous language.

c. If extension-field values are not composed entirely of graphic characters from the US-ASCII repertoire, a specification for how they are to be encoded as graphic US-ASCII characters in a Disposition-Notification-Options header field.

d. A reference to a permanent and readily available public specification that describes the semantics of the extension field.

11. Acknowledgements

The contributions of Bruce Lilly, Alfred Hoenes, Barry Leiba, Ben Campbell, Pete Resnick, Donald Eastlake and Alissa Cooper are gratefully acknowledged for this revision.

The contributions of Roger Fajman and Greg Vaudreuil to earlier versions of this document are also gratefully acknowledged.

12. References

12.1. Normative References

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <http://www.rfc-editor.org/info/rfc5321>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <http://www.rfc-editor.org/info/rfc5322>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <http://www.rfc-editor.org/info/rfc2045>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <http://www.rfc-editor.org/info/rfc2046>.

[RFC2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, DOI 10.17487/RFC2047, November 1996, <http://www.rfc-editor.org/info/rfc2047>.

[RFC6522] Kucherawy, M., Ed., "The Multipart/Report Media Type for the Reporting of Mail System Administrative Messages", STD 73, RFC 6522, DOI 10.17487/RFC6522, January 2012, <http://www.rfc-editor.org/info/rfc6522>.

[RFC3461] Moore, K., "Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)", RFC 3461, DOI 10.17487/RFC3461, January 2003, <http://www.rfc-editor.org/info/rfc3461>.

[RFC3464] Moore, K. and G. Vaudreuil, "An Extensible Message Format for Delivery Status Notifications", RFC 3464, DOI 10.17487/RFC3464, January 2003, <http://www.rfc-editor.org/info/rfc3464>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3503] Melnikov, A., "Message Disposition Notification (MDN) profile for Internet Message Access Protocol (IMAP)", RFC 3503, DOI 10.17487/RFC3503, March 2003, <http://www.rfc-editor.org/info/rfc3503>.

12.2. Informative References

[RFC2634] Hoffman, P., Ed., "Enhanced Security Services for S/MIME", RFC 2634, DOI 10.17487/RFC2634, June 1999, <http://www.rfc-editor.org/info/rfc2634>.

[RFC3249] Cancio, V., Moldovan, M., Tamura, H., and D. Wing, "Implementers Guide for Facsimile Using Internet Mail", RFC 3249, DOI 10.17487/RFC3249, September 2002, <http://www.rfc-editor.org/info/rfc3249>.

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, DOI 10.17487/RFC3501, March 2003, <http://www.rfc-editor.org/info/rfc3501>.

[RFC3801] Vaudreuil, G. and G. Parsons, "Voice Profile for Internet Mail - version 2 (VPIMv2)", RFC 3801, DOI 10.17487/RFC3801, June 2004, <http://www.rfc-editor.org/info/rfc3801>.

[RFC5233] Murchison, K., "Sieve Email Filtering: Subaddress Extension", RFC 5233, DOI 10.17487/RFC5233, January 2008, <http://www.rfc-editor.org/info/rfc5233>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5429] Stone, A., Ed., "Sieve Email Filtering: Reject and Extended Reject Extensions", RFC 5429, DOI 10.17487/RFC5429, March 2009, <http://www.rfc-editor.org/info/rfc5429>.

[RFC5598] Crocker, D., "Internet Mail Architecture", RFC 5598, DOI 10.17487/RFC5598, July 2009, <http://www.rfc-editor.org/info/rfc5598>.

[RFC6533] Hansen, T., Ed., Newman, C., and A. Melnikov, "Internationalized Delivery Status and Disposition Notifications", RFC 6533, DOI 10.17487/RFC6533, February 2012, <http://www.rfc-editor.org/info/rfc6533>.

Appendix A. Changes from RFC 3798

Changed IANA registration for different subregistries to "Specification Required" to match what is already used by IANA.

Updated IANA registration template for message/disposition-notification.

"X-" fields no longer reserved for experimental use and can now be registered in compliance with RFC 6648.

Fixed the default MTA-name-type used in "MDN-Gateway" to be "dns".

Strengthen requirements on obtaining user consent in order to protect user privacy.

Removed discussion of using source routes with MDNs, as source route is a deprecated Email feature.

The values of "dispatched" and "processed" were lost from the ABNF for "disposition-type". (Erratum #691)

Because the warning disposition modifier was previously removed, warning-field has also been removed. (Erratum #692)

Because the failed disposition type was previously removed, failure-field has also been removed.

The ABNF for ua-name and ua-product included semi-colon, which could not be distinguished from *text in the production. The ua-name was restricted to not include semi-colon. Semi-colon can still appear in the ua-product.

Removed recommendation to include the MUA DNS host name in the "Reporting-UA" MDN field.

The ABNF did not indicate all places that whitespace was allowable, in particular folding whitespace, although all implementations allow whitespace and folding in the header fields just like any other RFC5322 [RFC5322]-formatted header field. There were also a number of places in the ABNF that inconsistently permitted comments and whitespace in one leg of the production and not another. The ABNF now specifies FWS and CFWS in several places that should have already been specified by the grammar.

Extension-field was defined in the collected grammar but not in the main text.

The comparison of mailboxes in Disposition-Notification-To to the Return-Path addr-spec was clarified.

The use of the grammar production "parameter" was confusing with the RFC2045 [RFC2045] production of the same name, as well as other uses of the same term. These have been clarified.

A clarification was added on the extent of the 7bit nature of MDNs.

Uses of the terms "may" and "might" were clarified.

A clarification was added on the order of the fields in the message/disposition-notification content.

Authors' Addresses

Tony Hansen (editor)
AT&T Laboratories
200 Laurel Ave. South
Middletown, NJ 07748
USA

Email: tony@att.com

Alexey Melnikov (editor)
Isode Ltd
14 Castle Mews
Hampton, Middlesex TW12 2NP
UK

Email: Alexey.Melnikov@isode.com

Individual submission
Internet-Draft
Obsoletes: 7001, 7410
(if approved)
Intended status: Standards Track
Expires: December 6, 2015

M. Kucherawy
June 4, 2015

Message Header Field for Indicating Message Authentication Status
draft-ietf-appsawg-rfc7001bis-11

Abstract

This document specifies a message header field called Authentication-Results for use with electronic mail messages to indicate the results of message authentication efforts. Any receiver-side software, such as mail filters or Mail User Agents (MUAs), can use this header field to relay that information in a convenient and meaningful way to users or to make sorting and filtering decisions.

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 6, 2015.

Table of Contents

1. Introduction

This document describes a header field called Authentication-Results for electronic mail messages that presents the results of a message authentication effort in a machine-readable format. The intent of the header field is to create a place to collect such data when message authentication mechanisms are in use so that a Mail User Agent (MUA) and downstream filters can make filtering decisions and/or provide a recommendation to the user as to the validity of the message's origin and possibly the safety and integrity of its content.

This document revises the original definition found in [RFC5451] based upon various authentication protocols in current use and incorporates errata logged since the publication of the original specification.

End users are not expected to be direct consumers of this header field. This header field is intended for consumption by programs that will then use such data or render it in a human-usable form.

This document specifies the format of this header field and discusses the implications of its presence or absence. However, it does not discuss how the data contained in the header field ought to be used, such as what filtering decisions are appropriate or how an MUA might render those results, as these are local policy and/or user interface design questions that are not appropriate for this document.

At the time of publication of this document, the following are published email authentication methods:

- Author Domain Signing Practices ([ADSP]) (Historic)
- SMTP Service Extension for Authentication ([AUTH])
- DomainKeys Identified Mail Signatures ([DKIM])
- Domain-based Message Authentication, Reporting and Conformance ([DMARC])
- Sender Policy Framework ([SPF])
- reverse IP address name validation ("iprev", defined in Section 3)
- Require-Recipient-Valid-Since Header Field and SMTP Service Extension ([RRVS])

- S/MIME Signature Verification ([SMIME-REG])
- Vouch By Reference ([VBR])
- DomainKeys ([DOMAINKEYS]) (Historic)
- Sender ID ([SENDERID]) (Experimental)

There exist registries for tokens used within this header field that refer to the specifications listed above. Section 6 describes the registries and their contents, and specifies the process by which entries are added or updated. It also updates the existing contents to match the current states of these specifications.

This specification is not intended to be restricted to domain-based authentication schemes, but the existing schemes in that family have proven to be a good starting point for implementations. The goal is to give current and future authentication schemes a common framework within which to deliver their results to downstream agents and discourage the creation of unique header fields for each.

Although SPF defined a header field called "Received-SPF" and the historic DomainKeys defined one called "DomainKey-Status" for this purpose, those header fields are specific to the conveyance of their respective results only and thus are insufficient to satisfy the requirements enumerated below. In addition, many SPF implementations have adopted the header field specified here at least as an option, and DomainKeys has been obsoleted by DKIM.

1.1. Purpose

The header field defined in this document is expected to serve several purposes:

1. Convey the results of various message authentication checks, which are applied by upstream filters and Mail Transfer Agents (MTAs) and then passed to MUAs and downstream filters within the same "trust domain". Such agents might wish to render those results to end users or to use those data to apply more or less stringent content checks based on authentication results;
2. Provide a common location within a message for this data;
3. Create an extensible framework for reporting new authentication methods as they emerge.

In particular, the mere presence of this header field does not mean its contents are valid. Rather, the header field is reporting

assertions made by one or more authentication schemes (supposedly) applied somewhere upstream. For an MUA or downstream filter to treat the assertions as actually valid, there must be an assessment of the trust relationship among such agents, the validating MTA, and the mechanism for conveying the information.

1.2. Trust Boundary

This document makes several references to the "trust boundary" of an administrative management domain (ADMD). Given the diversity among existing mail environments, a precise definition of this term isn't possible.

Simply put, a transfer from the producer of the header field to the consumer must occur within a context that permits the consumer to treat assertions by the producer as being reliable and accurate (trustworthy). How this trust is obtained is outside the scope of this document. It is entirely a local matter.

Thus, this document defines a "trust boundary" as the delineation between "external" and "internal" entities. Services that are internal -- within the trust boundary -- are provided by the ADMD's infrastructure for its users. Those that are external are outside of the authority of the ADMD. By this definition, hosts that are within a trust boundary are subject to the ADMD's authority and policies, independent of their physical placement or their physical operation. For example, a host within a trust boundary might actually be operated by a remote service provider and reside physically within its data center.

It is possible for a message to be evaluated inside a trust boundary but then depart and re-enter the trust boundary. An example might be a forwarded message such as a message/rfc822 attachment (see Multipurpose Internet Mail Extensions [MIME]) or one that is part of a multipart/digest. The details reported by this field cannot be trusted in that case. Thus, this field found within one of those media types is typically ignored.

1.3. Processing Scope

The content of this header field is meant to convey to message consumers that authentication work on the message was already done within its trust boundary, and those results are being presented. It is not intended to provide message parameters to consumers so that they can perform authentication protocols on their own.

1.4. Requirements

This document establishes no new requirements on existing protocols or servers.

In particular, this document establishes no requirement on MTAs to reject or filter arriving messages that do not pass authentication checks. The data conveyed by the specified header field's contents are for the information of MUAs and filters and are to be used at their discretion.

1.5. Definitions

This section defines various terms used throughout this document.

1.5.1. 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].

1.5.2. Security

"Guidelines for Writing RFC Text on Security Considerations" ([SECURITY]) discusses authentication and authorization and the conflation of the two concepts. The use of those terms within the context of recent message security work has given rise to slightly different definitions, and this document reflects those current usages, as follows:

- "Authorization" is the establishment of permission to use a resource or represent an identity. In this context, authorization indicates that a message from a particular ADMD arrived via a route the ADMD has explicitly approved.

- "Authentication" is the assertion of validity of a piece of data about a message (such as the sender's identity) or the message in its entirety.

As examples: SPF and Sender ID are authorization mechanisms in that they express a result that shows whether or not the ADMD that apparently sent the message has explicitly authorized the connecting Simple Mail Transfer Protocol ([SMTP]) client to relay messages on its behalf, but they do not actually validate any other property of the message itself. By contrast, DKIM is agnostic as to the routing of a message but uses cryptographic signatures to authenticate agents, assign (some) responsibility for the message (which implies authorization), and ensure that the listed portions of the message

were not modified in transit. Since the signatures are not tied to SMTP connections, they can be added by either the ADMD of origin, intermediate ADMDs (such as a mailing list server), other handling agents, or any combination.

Rather than create a separate header field for each class of solution, this proposal groups them both into a single header field.

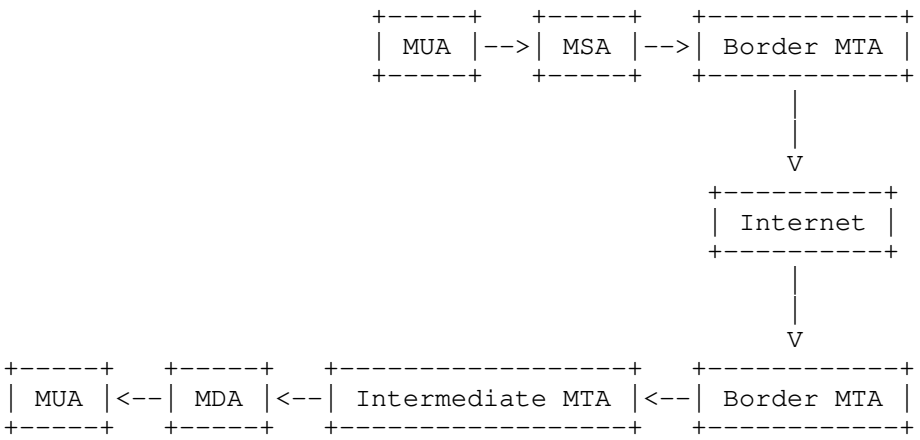
1.5.3. Email Architecture

- o A "border MTA" is an MTA that acts as a gateway between the general Internet and the users within an organizational boundary. (See also Section 1.2.)

- o A "delivery MTA" (or Mail Delivery Agent or MDA) is an MTA that actually enacts delivery of a message to a user's inbox or other final delivery.

- o An "intermediate MTA" is any MTA that is not a delivery MTA and is also not the first MTA to handle the message.

The following diagram illustrates the flow of mail among these defined components. See Internet Mail Architecture [EMAIL-ARCH] for further discussion on general email system architecture, which includes detailed descriptions of these components, and Appendix D of this document for discussion about the common aspects of email authentication in current environments.



Generally, it is assumed that the work of applying message authentication schemes takes place at a border MTA or a delivery MTA. This specification is written with that assumption in mind. However, there are some sites at which the entire mail infrastructure consists

of a single host. In such cases, such terms as "border MTA" and "delivery MTA" might well apply to the same machine or even the very same agent. It is also possible that some message authentication tests could take place on an intermediate MTA. Although this document doesn't specifically describe such cases, they are not meant to be excluded.

1.5.4. Other Terms

In this document, the term "producer" refers to any component that adds this header field to messages it is handling, and "consumer" refers to any component that identifies, extracts, and parses the header field to use as part of a handling decision.

1.6. Trust Environment

This header field permits one or more message validation mechanisms to communicate output to one or more separate assessment mechanisms. These mechanisms operate within a unified trust boundary that defines an Administrative Management Domain (ADMD). An ADMD contains one or more entities that perform validation and generate the header field and one or more that consume it for some type of assessment. The field often contains no integrity or validation mechanism of its own, so its presence must be trusted implicitly. Hence, valid use of the header field requires removing any occurrences of it that are present when the message enters the ADMD. This ensures that later occurrences have been added within the trust boundary of the ADMD.

The authserv-id token defined in Section 2.2 can be used to reference an entire ADMD or a specific validation engine within an ADMD. Although the labeling scheme is left as an operational choice, some guidance for selecting a token is provided in later sections of this document.

2. Definition and Format of the Header Field

This section gives a general overview of the format of the header field being defined and then provides more formal specification.

2.1. General Description

The header field specified here is called Authentication-Results. It is a Structured Header Field as defined in Internet Message Format ([MAIL]), and thus all of the related definitions in that document apply.

This header field is added at the top of the message as it transits MTAs that do authentication checks, so some idea of how far away the

checks were done can be inferred. It is therefore considered to be a trace field as defined in [MAIL], and thus all of the related definitions in that document apply.

The value of the header field (after removing comments) consists of an authentication identifier, an optional version, and then a series of statements and supporting data. The statements are of the form "method=result" and indicate which authentication method(s) were applied and their respective results. For each such statement, the supporting data can include a "reason" string and one or more "property=value" statements indicating which message properties were evaluated to reach that conclusion.

The header field can appear more than once in a single message, more than one result can be represented in a single header field, or a combination of these can be applied.

2.2. Formal Definition

Formally, the header field is specified as follows using Augmented Backus-Naur Form ([ABNF]):

```
authres-header = "Authentication-Results:" [CFWS] authserv-id
         [ CFWS authres-version ]
         ( no-result / 1*resinfo ) [CFWS] CRLF

authserv-id = value
            ; see below for a description of this element

authres-version = 1*DIGIT [CFWS]
        ; indicates which version of this specification is in use;
        ; this specification is version "1", and the absence of a
        ; version implies this version of the specification

no-result = [CFWS] ";" [CFWS] "none"
          ; the special case of "none" is used to indicate that no
          ; message authentication was performed

resinfo = [CFWS] ";" methodspec [ CFWS reasonspec ]
          *( CFWS propspec )

methodspec = [CFWS] method [CFWS] "=" [CFWS] result
           ; indicates which authentication method was evaluated
           ; and what its output was

reasonspec = "reason" [CFWS] "=" [CFWS] value
           ; a free-form comment on the reason the given result
           ; was returned
```

```
propspec = ptype [CFWS] "." [CFWS] property [CFWS] "=" pvalue
         ; an indication of which properties of the message
         ; were evaluated by the authentication scheme being
         ; applied to yield the reported result

method = Keyword [ [CFWS] "/" [CFWS] method-version ]
       ; a method indicates which method's result is
       ; represented by "result", and is one of the methods
       ; explicitly defined as valid in this document
       ; or is an extension method as defined below

method-version = 1*DIGIT [CFWS]
       ; indicates which version of the method specification is
       ; in use, corresponding to the matching entry in the IANA
       ; "Email Authentication Methods" registry; a value of "1"
       ; is assumed if this version string is absent

result = Keyword
       ; indicates the results of the attempt to authenticate
       ; the message; see below for details

ptype = Keyword
      ; indicates whether the property being evaluated was
      ; a parameter to an [SMTP] command, was a value taken
      ; from a message header field, was some property of
      ; the message body, or was some other property evaluated by
      ; the receiving MTA; expected to be one of the "property
      ; types" explicitly defined as valid, or an extension
      ; ptype, as defined below

property = special-smtp-verb / Keyword
        ; indicates more specifically than "ptype" what the
        ; source of the evaluated property is; the exact meaning
        ; is specific to the method whose result is being reported
        ; and is defined more clearly below

special-smtp-verb = "mailfrom" / "rcptto"
        ; special cases of [SMTP] commands that are made up
        ; of multiple words

pvalue = [CFWS] ( value / [ [ local-part ] "@" ] domain-name )
         [CFWS]
       ; the value extracted from the message property defined
       ; by the "ptype.property" construction
```

"local-part" is defined in Section 3.4.1 of [MAIL], and "CFWS" is defined in Section 3.2.2 of [MAIL].

"Keyword" is defined in Section 4.1.2 of [SMTP].

The "value" is as defined in Section 5.1 of [MIME].

The "domain-name" is as defined in Section 3.5 of [DKIM].

The "Keyword" used in "result" above is further constrained by the necessity of being enumerated in Section 2.7.

See Section 2.5 for a description of the authserv-id element.

If the value portion of a "pvalue" construction identifies something intended to be an e-mail identity, then it MUST use the right hand portion of that ABNF definition.

The list of commands eligible for use with the "smtp" ptype can be found in Section 4.1 of [SMTP].

The "propspec" may be omitted if, for example, the method was unable to extract any properties to do its evaluation yet has a result to report.

Where an SMTP command name is being reported as a "property", the agent generating the header field represents that command by converting it to lowercase and dropping any spaces (e.g., "MAIL FROM" becomes "mailfrom", "RCPT TO" becomes "rcptto", etc.).

A "ptype" value of "policy" indicates a policy decision about the message not specific to a property of the message that could be extracted. See Section 2.4 for details.

Examples of complete messages using this header field can be found in Appendix C.

2.3. Property Types (ptypes) and Properties

The "ptype" in the ABNF above indicates the general type of property being described by the result being reported, upon which the reported result was based. Coupled with the "property", which is more specific, they indicate from which particular part of the message the reported data were extracted.

Combinations of ptypes and properties are registered and described in the "Email Authentication Methods" registry, coupled with the authentication methods with which they are used. This is further described in Section 6.

Legal values of "ptype" are as defined in the IANA "Email

Authentication Property Types" registry, created by [PTYPES-REGISTRY]. The initial values and what they typically indicate are as follows, copied from [RFC7001]:

body: Information that was extracted from the body of the message. This might be an arbitrary string of bytes, a hash of a string of bytes, a Uniform Resource Identifier, or some other content of interest. The "property" is an indication of where within the message body the extracted content was found, and can indicate an offset, identify a MIME part, etc.

header: Indicates information that was extracted from the header of the message. This might be the value of a header field or some portion of a header field. The "property" gives a more precise indication of the place in the header from which the extraction took place.

policy: A local policy mechanism was applied that augments or overrides the result returned by the authentication mechanism. (See Section 2.4.)

smtp: Indicates information that was extracted from an SMTP command that was used to relay the message. The "property" indicates which SMTP command included the extracted content as a parameter.

Results reported using unknown ptypes MUST NOT be used in making handling decisions. They can be safely ignored by consumers.

Entries in the "Email Authentication Methods" registry can define properties that deviate from these definitions when appropriate. Such deviations need to be clear in the registry and/or in the defining document. See Section 2.7.1 for an example.

2.4. The "policy" ptype

A special ptype value of "policy" is also defined. This ptype is provided to indicate that some local policy mechanism was applied that augments or even replaces (i.e., overrides) the result returned by the authentication mechanism. The property and value in this case identify the local policy that was applied and the result it returned.

For example, a DKIM signature is not required to include the Subject header field in the set of fields that are signed. An ADMD receiving such a message might decide that such a signature is unacceptable, even if it passes, because the content of the Subject header field could be altered post-signing without invalidating the signature. Such an ADMD could replace the DKIM "pass" result with a "policy"

result and then also include the following in the corresponding Authentication-Result field:

... dkim=fail policy.dkim-rules=unsigned-subject ...

In this case, the property is "dkim-rules", indicating some local check by that name took place and that check returned a result of "unsigned-subject". These are arbitrary names selected by (and presumably used within) the ADMD making use of them, so they are not normally registered with IANA or otherwise specified apart from setting syntax restrictions that allow for easy parsing within the rest of the header field.

This ptype existed in the original specification for this header field, but without a complete description or example of intended use. As a result, it has not seen any practical use to date that matches its intended purpose. These added details are provided to guide implementers toward proper use.

2.5. Authentication Identifier Field

Every Authentication-Results header field has an authentication service identifier field (authserv-id above). Specifically, this is any string intended to identify the authentication service within the ADMD that conducted authentication checks on the message. This identifier is intended to be machine-readable and not necessarily meaningful to users.

Since agents consuming this field will use this identifier to determine whether its contents are of interest (and are safe to use), the uniqueness of the identifier MUST be guaranteed by the ADMD that generates it and MUST pertain to that ADMD. MUAs or downstream filters SHOULD use this identifier to determine whether or not the data contained in an Authentication-Results header field ought to be used or ignored.

For simplicity and scalability, the authentication service identifier SHOULD be a common token used throughout the ADMD. Common practice is to use the DNS domain name used by or within that ADMD, sometimes called the "organizational domain", but this is not strictly necessary.

For tracing and debugging purposes, the authentication identifier can instead be the specific hostname of the MTA performing the authentication check whose result is being reported. Moreover, some implementations define a substructure to the identifier; these are outside of the scope of this specification.

Note, however, that using a local, relative identifier like a flat hostname, rather than a hierarchical and globally unique ADMD identifier like a DNS domain name, makes configuration more difficult for large sites. The hierarchical identifier permits aggregating related, trusted systems together under a single, parent identifier, which in turn permits assessing the trust relationship with a single reference. The alternative is a flat namespace requiring individually listing each trusted system. Since consumers will use the identifier to determine whether to use the contents of the header field:

- Changes to the identifier impose a large, centralized administrative burden.

- Ongoing administrative changes require constantly updating this centralized table, making it difficult to ensure that an MUA or downstream filter will have access to accurate information for assessing the usability of the header field's content. In particular, consumers of the header field will need to know not only the current identifier(s) in use but previous ones as well to account for delivery latency or later re-assessment of the header field's contents.

Examples of valid authentication identifiers are "example.com", "mail.example.org", "ms1.newyork.example.com", and "example-auth".

2.6. Version Tokens

The grammar above provides for the optional inclusion of versions on both the header field itself (attached to the authserv-id token) and on each of the methods being reported. The method version refers to the method itself, which is specified in the documents describing those methods, while the authserv-id version refers to this document and thus the syntax of this header field.

The purpose of including these is to avoid misinterpretation of the results. That is, if a parser finds a version after an authserv-id that it does not explicitly know, it can immediately discontinue trying to parse since what follows might not be in an expected format. For a method version, the parser SHOULD ignore a method result if the version is not supported in case the semantics of the result have a different meaning than what is expected. For example, if a hypothetical DKIM version 2 yielded a "pass" result for different reasons than version 1 does, a consumer of this field might not want to use the altered semantics. Allowing versions in the syntax is a way to indicate this and let the consumer of the header field decide.

2.7. Defined Methods and Result Values

Each individual authentication method returns one of a set of specific result values. The subsections below provide references to the documents defining the authentication methods specifically supported by this document, and their corresponding result values. Verifiers SHOULD use these values as described below. New methods not specified in this document, but intended to be supported by the header field defined here, MUST include a similar result table either in their defining documents or in supplementary ones.

2.7.1. DKIM and DomainKeys

DKIM is represented by the "dkim" method and is defined in [DKIM]. DomainKeys is defined in [DOMAINKEYS] and is represented by the "domainkeys" method.

Section 3.8 of [DOMAINKEYS] enumerates some possible results of a DomainKeys evaluation. Those results are not used when generating this header field; rather, the results returned are listed below.

A signature is "acceptable to the ADMD" if it passes local policy checks (or there are no specific local policy checks). For example, an ADMD policy might require that the signature(s) on the message be added using the DNS domain present in the From header field of the message, thus making third-party signatures unacceptable even if they verify.

Both DKIM and DomainKeys use the same result set, as follows:

none: The message was not signed.

pass: The message was signed, the signature or signatures were acceptable to the ADMD, and the signature(s) passed verification tests.

fail: The message was signed and the signature or signatures were acceptable to the ADMD, but they failed the verification test(s).

policy: The message was signed, but some aspect of the signature or signatures was not acceptable to the ADMD.

neutral: The message was signed, but the signature or signatures contained syntax errors or were not otherwise able to be processed. This result is also used for other failures not covered elsewhere in this list.

temperror: The message could not be verified due to some error that is likely transient in nature, such as a temporary inability to retrieve a public key. A later attempt may produce a final result.

permerror: The message could not be verified due to some error that is unrecoverable, such as a required header field being absent. A later attempt is unlikely to produce a final result.

DKIM results are reported using a ptype of "header". The property, however, represents one of the tags found in the DKIM-Signature header field rather than a distinct header field. For example, the ptype-property combination "header.d" refers to the content of the "d" (signing domain) tag from within the signature header field, and not a distinct header field called "d".

The ability to report different DKIM results for a multiply-signed message is described in [RFC6008].

[DKIM] advises that if a message fails verification, it is to be treated as an unsigned message. A report of "fail" here permits the receiver of the report to decide how to handle the failure. A report of "neutral" or "none" preempts that choice, ensuring the message will be treated as if it had not been signed.

Section 3.1 of [DOMAINKEYS] describes a process by which the sending address of the message is determined. DomainKeys results are thus reported along with the signing domain name, the sending address of the message, and the name of the header field from which the latter was extracted. This means that a DomainKeys result includes a ptype-property combination of "header.d", plus one of "header.from" and "header.sender". The sending address extracted from the header is included with any [MAIL]-style comments removed; moreover, the local-part of the address and the "@" character are removed if it has not been authenticated in some way.

2.7.2. SPF and Sender ID

SPF and Sender ID use the "spf" and "sender-id" method names, respectively. The result values for SPF are defined in Section 2.6 of [SPF], and those definitions are included here by reference:

Code	Meaning
none	[RFC7208], Section 2.6.1
pass	[RFC7208], Section 2.6.3
fail	[RFC7208], Section 2.6.4
softfail	[RFC7208], Section 2.6.5
policy	[this RFC], Section 2.4
neutral	[RFC7208], Section 2.6.2
temperror	[RFC7208], Section 2.6.6
permerror	[RFC7208], Section 2.6.7

These result codes are used in the context of this specification to reflect the result returned by the component conducting SPF evaluation.

For SPF, the ptype used is "smtp", and the property is either "mailfrom" or "helo", since those values are the ones SPF can evaluate. (If the SMTP client issued the EHLO command instead of HELO, the property used is "helo".)

The "sender-id" method is described in [SENDERID]. For this method, the ptype used is "header" and the property will be the name of the header field from which the Purported Responsible address (see [PRA]) was extracted, namely one of "Resent-Sender", "Resent-From", "Sender", or "From".

The results for Sender ID are listed and described in Section 4.2 of [SENDERID], but for the purposes of this specification, the SPF definitions enumerated above are used instead. Also, [SENDERID] specifies result codes that use mixed case, but they are typically used all lowercase in this context.

For both methods, an additional result of "policy" is defined, which means the client was authorized to inject or relay mail on behalf of the sender's DNS domain according to the authentication method's algorithm, but local policy dictates that the result is unacceptable. For example, "policy" might be used if SPF returns a "pass" result, but a local policy check matches the sending DNS domain to one found in an explicit list of unacceptable DNS domains (e.g., spammers).

If the retrieved sender policies used to evaluate SPF and Sender ID do not contain explicit provisions for authenticating the local-part (see Section 3.4.1 of [MAIL]) of an address, the "pvalue" reported along with results for these mechanisms SHOULD NOT include the local-part or the following "@" character.

2.7.3. "iprev"

The result values used by the "iprev" method, defined in Section 3, are as follows:

pass: The DNS evaluation succeeded, i.e., the "reverse" and "forward" lookup results were returned and were in agreement.

fail: The DNS evaluation failed. In particular, the "reverse" and "forward" lookups each produced results, but they were not in agreement, or the "forward" query completed but produced no result, e.g., a DNS RCODE of 3, commonly known as NXDOMAIN, or an RCODE of 0 (NOERROR) in a reply containing no answers, was returned.

temperror: The DNS evaluation could not be completed due to some error that is likely transient in nature, such as a temporary DNS error, e.g., a DNS RCODE of 2, commonly known as SERVFAIL, or other error condition resulted. A later attempt may produce a final result.

permerror: The DNS evaluation could not be completed because no PTR data are published for the connecting IP address, e.g., a DNS RCODE of 3, commonly known as NXDOMAIN, or an RCODE of 0 (NOERROR) in a reply containing no answers, was returned. This prevented completion of the evaluation. A later attempt is unlikely to produce a final result.

There is no "none" for this method since any TCP connection delivering email has an IP address associated with it, so some kind of evaluation will always be possible.

The result is reported using a ptype of "policy" (as this is not part of any established protocol) and a property of "iprev".

For discussion of the format of DNS replies, see "Domain Names - Implementation and Specification" ([DNS]).

2.7.4. SMTP AUTH

SMTP AUTH (defined in [AUTH]) is represented by the "auth" method. Its result values are as follows:

none: SMTP authentication was not attempted.

pass: The SMTP client authenticated to the server reporting the result using the protocol described in [AUTH].

fail: The SMTP client attempted to authenticate to the server using the protocol described in [AUTH] but was not successful (such as providing a valid identity but an incorrect password).

temperror: The SMTP client attempted to authenticate using the protocol described in [AUTH] but was not able to complete the attempt due to some error that is likely transient in nature, such as a temporary directory service lookup error. A later attempt may produce a final result.

permerror: The SMTP client attempted to authenticate using the protocol described in [AUTH] but was not able to complete the attempt due to some error that is likely not transient in nature, such as a permanent directory service lookup error. A later attempt is not likely to produce a final result.

The result of AUTH is reported using a ptype of "smtp" and a property of either:

- "auth", in which case the value is the authorization identity generated by the exchange initiated by the AUTH command; or

- "mailfrom", in which case the value is the mailbox identified by the AUTH parameter used with the MAIL FROM command.

If both identities are available, both can be reported. For example, consider this command issued by a client that has completed session authentication with the AUTH command resulting in an authorized identity of "client@c.example":

```
MAIL FROM:<alice@a.example> AUTH=<bob@b.example>
```

This could result in a resinfo construction like so:

```
; auth=pass smtp.auth=client@c.example smtp.mailfrom=bob@b.example
```

Note that in all cases other than "pass", the message was sent by an unauthenticated client. All non-"pass" cases SHOULD thus be treated as equivalent with respect to this method.

2.7.5. Other Registered Codes

Result codes were also registered in other RFCs as follows:

- Vouch By Reference (in [AR-VBR], represented by "vbr");
- Authorized Third-Party Signatures (in [ATPS], represented by "dkim-atps");
- Author Domain Signing Practices (in [ADSP], represented by "dkim-adsp");
- Require-Recipient-Valid-Since (in [RRVS], represented by "rrvs");
- S/MIME (in [SMIME-REG], represented by "smime").

2.7.6. Extension Methods

Additional authentication method identifiers (extension methods) may be defined in the future by later revisions or extensions to this specification. These method identifiers are registered with the Internet Assigned Numbers Authority (IANA) and, preferably, published in an RFC. See Section 6 for further details.

Extension methods can be defined for the following reasons:

1. To allow additional information from new authentication systems to be communicated to MUAs or downstream filters. The names of such identifiers ought to reflect the name of the method being defined but ought not be needlessly long.
2. To allow the creation of "sub-identifiers" that indicate different levels of authentication and differentiate between their relative strengths, e.g., "auth1-weak" and "auth1-strong".

Authentication method implementers are encouraged to provide adequate information, via message header field comments if necessary, to allow an MUA developer to understand or relay ancillary details of authentication results. For example, if it might be of interest to relay what data was used to perform an evaluation, such information could be relayed as a comment in the header field, such as:

```
Authentication-Results: example.com;
          foo=pass bar.baz=blob (2 of 3 tests OK)
```

Experimental method identifiers MUST only be used within ADMDs that have explicitly consented to use them. These method identifiers and the parameters associated with them are not documented in RFCs.

Therefore, they are subject to change at any time and not suitable for production use. Any MTA, MUA, or downstream filter intended for production use SHOULD ignore or delete any Authentication-Results header field that includes an experimental (unknown) method identifier.

2.7.7. Extension Result Codes

Additional result codes (extension results) might be defined in the future by later revisions or extensions to this specification. Result codes MUST be registered with the Internet Assigned Numbers Authority (IANA) and preferably published in an RFC. See Section 6 for further details.

Experimental results MUST only be used within ADMDs that have explicitly consented to use them. These results and the parameters associated with them are not formally documented. Therefore, they are subject to change at any time and not suitable for production use. Any MTA, MUA, or downstream filter intended for production use SHOULD ignore or delete any Authentication-Results header field that includes an extension result.

3. The "iprev" Authentication Method

This section defines an additional authentication method called "iprev".

"iprev" is an attempt to verify that a client appears to be valid based on some DNS queries, which is to say that the IP address is explicitly associated with a domain name. Upon receiving a session initiation of some kind from a client, the IP address of the client peer is queried for matching names (i.e., a number-to-name translation, also known as a "reverse lookup" or a "PTR" record query). Once that result is acquired, a lookup of each of the names (i.e., a name-to-number translation, or an "A" or "AAAA" record query) thus retrieved is done. The response to this second check will typically result in at least one mapping back to the client's IP address.

Expressed as an algorithm: If the client peer's IP address is I, the list of names to which I maps (after a "PTR" query) is the set N, and the union of IP addresses to which each member of N maps (after corresponding "A" and "AAAA" queries) is L, then this test is successful if I is an element of L.

Often an MTA receiving a connection that fails this test will simply reject the connection using the enhanced status code defined in [AUTH-ESC]. If an operator instead wishes to make this information

available to downstream agents as a factor in handling decisions, it records a result in accordance with Section 2.7.3.

The response to a PTR query could contain multiple names. To prevent heavy DNS loads, agents performing these queries MUST be implemented such that the number of names evaluated by generation of corresponding A or AAAA queries is limited so as not to be unduly taxing to the DNS infrastructure, though it MAY be configurable by an administrator. As an example, Section 4.6.4 of [SPF] chose a limit of 10 for its implementation of this algorithm.

"DNS Extensions to Support IP Version 6" ([DNS-IP6]) discusses the query formats for the IPv6 case.

There is some contention regarding the wisdom and reliability of this test. For example, in some regions, it can be difficult for this test ever to pass because the practice of arranging to match the forward and reverse DNS is infrequently observed. Therefore, the precise implementation details of how a verifier performs an "iprev" test are not specified here. The verifier MAY report a successful or failed "iprev" test at its discretion having done some kind of check of the validity of the connection's identity using DNS. It is incumbent upon an agent making use of the reported "iprev" result to understand what exactly that particular verifier is attempting to report.

Extensive discussion of reverse DNS mapping and its implications can be found in "Considerations for the use of DNS Reverse Mapping" ([DNSOP-REVERSE]). In particular, it recommends that applications avoid using this test as a means of authentication or security. Its presence in this document is not an endorsement but is merely acknowledgment that the method remains common and provides the means to relay the results of that test.

4. Adding the Header Field to a Message

This specification makes no attempt to evaluate the relative strengths of various message authentication methods that may become available. The methods listed are an order-independent set; their sequence does not indicate relative strength or importance of one method over another. Instead, the MUA or downstream filter consuming this header field is to interpret the result of each method based on its own knowledge of what that method evaluates.

Each "method" MUST refer to an authentication method declared in the IANA registry or an extension method as described in Section 2.7.6, and each "result" MUST refer to a result code declared in the IANA registry or an extension result code as defined in Section 2.7.7.

See Section 6 for further information about the registered methods and result codes.

An MTA compliant with this specification adds this header field (after performing one or more message authentication tests) to indicate which MTA or ADMD performed the test, which test got applied, and what the result was. If an MTA applies more than one such test, it adds this header field either once per test or once indicating all of the results. An MTA MUST NOT add a result to an existing header field.

An MTA MAY add this header field containing only the authentication identifier portion and the "none" token (see Section 2.2) to indicate explicitly that no message authentication schemes were applied prior to delivery of this message.

An MTA adding this header field has to take steps to identify it as legitimate to the MUAs or downstream filters that will ultimately consume its content. One process to do so is described in Section 5. Further measures may be necessary in some environments. Some possible solutions are enumerated in Section 7.1. This document does not mandate any specific solution to this issue as each environment has its own facilities and limitations.

Most known message authentication methods focus on a particular identifier to evaluate. SPF and Sender ID differ in that they can yield a result based on more than one identifier; specifically, SPF can evaluate the RFC5321.HELO parameter or the RFC5321.MailFrom parameter, and Sender ID can evaluate the RFC5321.MailFrom parameter or the Purported Responsible Address (PRA) identity. When generating this field to report those results, only the parameter that yielded the result is included.

For MTAs that add this header field, adding header fields in order (at the top), per Section 3.6 of [MAIL], is particularly important. Moreover, this header field SHOULD be inserted above any other trace header fields such MTAs might prepend. This placement allows easy detection of header fields that can be trusted.

End users making direct use of this header field might inadvertently trust information that has not been properly vetted. If, for example, a basic SPF result were to be relayed that claims an authenticated addr-spec, the local-part of that addr-spec has actually not been authenticated. Thus, an MTA adding this header field SHOULD NOT include any data that has not been authenticated by the method(s) being applied. Moreover, MUAs SHOULD NOT render to users such information if it is presented by a method known not to authenticate it.

4.1. Header Field Position and Interpretation

In order to ensure non-ambiguous results and avoid the impact of false header fields, MUAs and downstream filters SHOULD NOT interpret this header field unless specifically configured to do so by the user or administrator. That is, this interpretation should not be "on by default". Naturally then, users or administrators ought not activate such a feature unless they are certain the header field will be validly added by an agent within the ADMD that accepts the mail that is ultimately read by the MUA, and instances of the header field appearing to originate within the ADMD but are actually added by foreign MTAs will be removed before delivery.

Furthermore, MUAs and downstream filters SHOULD NOT interpret this header field unless the authentication service identifier it bears appears to be one used within its own ADMD as configured by the user or administrator.

MUAs and downstream filters MUST ignore any result reported using a "result" not specified in the IANA "Result Code" registry or a "ptype" not listed in the corresponding registry for such values as defined in Section 6. Moreover, such agents MUST ignore a result indicated for any "method" they do not specifically support.

An MUA SHOULD NOT reveal these results to end users, absent careful human factors design considerations and testing, for the presentation of trust-related materials. For example, an attacker could register example.com (note the digit "one") and send signed mail to intended victims; a verifier would detect that the signature was valid and report a "pass" even though it's clear the DNS domain name was intended to mislead. See Section 7.2 for further discussion.

As stated in Section 2.1, this header field MUST be treated as though it were a trace header field as defined in Section 3.6.7 of [MAIL] and hence MUST NOT be reordered and MUST be prepended to the message, so that there is generally some indication upon delivery of where in the chain of handling MTAs the message authentication was done.

Note that there are a few message handlers that are only capable of appending new header fields to a message. Strictly speaking, these handlers are not compliant with this specification. They can still add the header field to carry authentication details, but any signal about where in the handling chain the work was done may be lost. Consumers SHOULD be designed such that this can be tolerated, especially from a producer known to have this limitation.

MUAs SHOULD ignore instances of this header field discovered within message/rfc822 MIME attachments.

Further discussion of these topics can be found in Section 7 below.

4.2. Local Policy Enforcement

Some sites have a local policy that considers any particular authentication policy's non-recoverable failure results (typically "fail" or similar) as justification for rejecting the message. In such cases, the border MTA SHOULD issue an SMTP rejection response to the message, rather than adding this header field and allowing the message to proceed toward delivery. This is more desirable than allowing the message to reach an internal host's MTA or spam filter, thus possibly generating a local rejection such as a Delivery Status Notification (DSN) [DSN] to a forged originator. Such generated rejections are colloquially known as "backscatter".

The same MAY also be done for local policy decisions overriding the results of the authentication methods (e.g., the "policy" result codes described in Section 2.7).

Such rejections at the SMTP protocol level are not possible if local policy is enforced at the MUA and not the MTA.

5. Removing Existing Header Fields

For security reasons, any MTA conforming to this specification MUST delete any discovered instance of this header field that claims, by virtue of its authentication service identifier, to have been added within its trust boundary but that did not come directly from another trusted MTA. For example, an MTA for example.com receiving a message MUST delete or otherwise obscure any instance of this header field bearing an authentication service identifier indicating that the header field was added within example.com prior to adding its own header fields. This could mean each MTA will have to be equipped with a list of internal MTAs known to be compliant (and hence trustworthy).

For simplicity and maximum security, a border MTA could remove all instances of this header field on mail crossing into its trust boundary. However, this may conflict with the desire to access authentication results performed by trusted external service providers. It may also invalidate signed messages whose signatures cover external instances of this header field. A more robust border MTA could allow a specific list of authenticating MTAs whose information is to be admitted, removing the header field originating from all others.

As stated in Section 1.2, a formal definition of "trust boundary" is deliberately not made here. It is entirely possible that a border

MTA for example.com will explicitly trust authentication results asserted by upstream host example.net even though they exist in completely disjoint administrative boundaries. In that case, the border MTA MAY elect not to delete those results; moreover, the upstream host doing some authentication work could apply a signing technology such as [DKIM] on its own results to assure downstream hosts of their authenticity. An example of this is provided in Appendix C.

Similarly, in the case of messages signed using [DKIM] or other message-signing methods that sign header fields, this removal action could invalidate one or more signatures on the message if they covered the header field to be removed. This behavior can be desirable since there's little value in validating the signature on a message with forged header fields. However, signing agents MAY therefore elect to omit these header fields from signing to avoid this situation.

An MTA SHOULD remove any instance of this header field bearing a version (express or implied) that it does not support. However, an MTA MUST remove such a header field if the [SMTP] connection relaying the message is not from a trusted internal MTA. This means the MTA needs to be able to understand versions of this header field at least as late as the ones understood by the MUAs or other consumers within its ADMD.

6. IANA Considerations

IANA has registered the defined header field and created tables as described below. These registry actions were originally defined by [RFC5451] and updated by [RFC6577] and [RFC7001]. The created registries are being further updated here to increase their completeness.

6.1. The Authentication-Results Header Field

[RFC5451] added the Authentication-Results header field to the IANA "Permanent Message Header Field Names" registry, per the procedure found in [IANA-HEADERS]. That entry is to be updated to reference this document. The following is the registration template:

Header field name: Authentication-Results
Applicable protocol: mail ([MAIL])
Status: Standard
Author/Change controller: IETF
Specification document(s): [this RFC]
Related information: none

6.2. "Email Authentication Methods" Registry Description

Names of message authentication methods supported by this specification are to be registered with IANA, with the exception of experimental names as described in Section 2.7.6. Along with each method is recorded the properties that accompany the method's result.

The "Email Authentication Parameters" group, and within it the "Email Authentication Methods" registry, were created by [RFC5451] for this purpose. [RFC6577] added a "status" field for each entry. [RFC7001] amended the rules governing that registry, and also added a "version" field to the registry.

The reference for that registry shall be updated to reference this document.

New entries are assigned only for values that have received Expert Review, per [IANA-CONSIDERATIONS]. The designated expert shall be appointed by the IESG. The designated expert has discretion to request that a publication be referenced if a clear, concise definition of the authentication method cannot be provided such that interoperability is assured. Registrations should otherwise be permitted. The designated expert can also handle requests to mark any current registration as "deprecated".

No two entries can have the same combination of method, ptype, and property.

An entry in this registry contains the following:

Method: the name of the method;

Definition: a reference to the document that created this entry, if any (see below);

ptype: a "ptype" value appropriate for use with that method;

property: a "property" value matching that "ptype" also appropriate for use with that method;

Value: a brief description of the value to be supplied with that method/ptype/property tuple;

Status: the status of this entry, which is either:

active: The entry is in current use.

deprecated: The entry is no longer in current use.

Version: a version number associated with the method (preferably starting at "1").

The "Definition" field will typically refer to a permanent document, or at least some descriptive text, where additional information about the entry being added can be found. This might in turn reference the document where the method is defined so that all of the semantics around creating or interpreting an Authentication-Results header field using this method, ptype, and property can be understood.

6.3. "Email Authentication Methods" Registry Update

The following changes are to be made to this registry upon approval of this document:

1. The "Defined" field shall be renamed to "Definition", to be consistent with the other registries in this group.

2. The current entry for the "auth" method shall have its "property" field changed to "mailfrom", and its "Definition" field changed to this document.

3. The entry for the "dkim" method, "header" ptype and "b" property shall now reference [RFC6008] as its defining document, and the reference shall be removed from the description.

4. All other "dkim", "domainkeys", "iprev", "sender-id", and "spf" method entries shall have their "Definition" fields changed to this document, as this document contains a complete description of the registry and these corresponding values.

5. All "smime" entries have their "Definition" fields changed to [SMIME-REG].

6. The "value" field of the "smime" entry using property "smime-part" shall be changed to read: "The MIME body part reference that contains the S/MIME signature. See Section 3.2.1 of RFC7281 for full syntax."

7. The following entry is to be added:

Method: auth

Definition: [this document]

ptype: smtp

property: auth

Value: identity confirmed by the AUTH command

Status: active

Version: 1

8. The values of the "domainkeys" entries for ptype "header" are updated as follows:

   from: contents of the [MAIL] From: header field, after removing comments, and removing the local-part and following "@" if not authenticated

   sender: contents of the [MAIL] Sender: header field, after removing comments, and removing the local-part and following "@" if not authenticated

9. All entries for "dkim-adsp" and "domainkeys" shall have their Status values changed to "deprecated", reflecting the fact that the corresponding specifications now have Historical status. Their "Definition" fields shall also be modified to include a reference to this document.

6.4. "Email Authentication Property Types" Registry

[PTYPES-REGISTRY] created the Email Authentication Property Types registry.

Entries in this registry are subject to the Expert Review rules as described in [IANA-CONSIDERATIONS]. Each entry in the registry requires the following values:

ptype: The name of the ptype being registered, which must fit within the ABNF described in Section 2.2.

Definition: An optional reference to a defining specification.

Description: A brief description of what sort of information this "ptype" is meant to cover.

For new entries, the Designated Expert needs to assure that the description provided for the new entry adequately describes the intended use. An example would be helpful to include in the entry's defining document, if any, although entries in the "Email Authentication Methods" registry or the "Email Authentication Result Names" registry might also serve as examples of intended use.

As this is a complete re-statement of the definition and rules for this registry, IANA shall update this registry to show Section 2.3 of this document as the current definitions for the "body", "header", "policy" and "smtp" entries of that registry. References to [RFC7001] shall be removed.

6.5. "Email Authentication Result Names" Description

Names of message authentication result codes supported by this specification must be registered with IANA, with the exception of experimental codes as described in Section 2.7.7. A registry was created by [RFC5451] for this purpose. [RFC6577] added the "status" column, and [RFC7001] updated the rules governing that registry.

New entries are assigned only for values that have received Expert Review, per [IANA-CONSIDERATIONS]. The designated expert shall be appointed by the IESG. The designated expert has discretion to request that a publication be referenced if a clear, concise definition of the authentication result cannot be provided such that interoperability is assured. Registrations should otherwise be permitted. The designated expert can also handle requests to mark any current registration as "deprecated".

No two entries can have the same combination of method and code.

An entry in this registry contains the following:

Auth Method: an authentication method for which results are being returned using the header field defined in this document;

Code: a result code that can be returned for this authentication method;

Specification: either free form text explaining the meaning of this method-code combination, or a reference to such a definition.

Status: the status of this entry, which is either:

active: The entry is in current use.

deprecated: The entry is no longer in current use.

6.6. "Email Authentication Result Names" Update

This document includes a complete description of the registry, obsoleting [RFC7001]. Accordingly, the following changes are to be made to this registry on publication of this document:

- The "Defined" field shall be removed.
- The "Meaning" field shall be renamed to "Specification", as described above.
- The "Auth Method" field shall appear before the "Code" field.
- For easier searching, the table shall be arranged such that it is sorted first by Auth Method, then by Code within each Auth Method grouping.
- All entries for the "dkim", "domainkeys", "spf", "sender-id", "auth", and "iprev" methods shall have their "Specification" fields replaced as follows:

  dkim: [this document] Section 2.7.1

  domainkeys: [this document] Section 2.7.1

  spf: for "hardfail", [RFC5451] Section 2.4.2; for all others, [this document] Section 2.7.2

  sender-id: for "hardfail", [RFC5451] Section 2.4.2; for all others, [this document] Section 2.7.2

  auth: [this document] Section 2.7.4

  iprev: [this document] Section 2.7.3

- All entries for "dkim-adsp" that are missing an explicit reference to a defining document shall have [ADSP] added to their "Specification" fields.
- All entries for "dmarc" shall have their "Specification" fields changed to reference Secton 11.2 of [DMARC].

- All entries for "dkim-adsp" and "domainkeys" shall have their Status values changed to "deprecated", reflecting the fact that the corresponding specifications now have Historical status. Their "Specification" fields shall also be modified to include a reference to this document.

6.7. SMTP Enhanced Status Codes

The entry for X.7.25 in the Enumerated Status Codes sub-registry of the SMTP Enhanced Status Codes registry is to be updated to refer to this document instead of [RFC7001].

7. Security Considerations

The following security considerations apply when adding or processing the Authentication-Results header field:

7.1. Forged Header Fields

An MUA or filter that accesses a mailbox whose messages are handled by a non-conformant MTA, and understands Authentication-Results header fields, could potentially make false conclusions based on forged header fields. A malicious user or agent could forge a header field using the DNS domain of a receiving ADMD as the authserv-id token in the value of the header field and, with the rest of the value, claim that the message was properly authenticated. The non-conformant MTA would fail to strip the forged header field, and the MUA could inappropriately trust it.

For this reason, it is best not to have processing of the Authentication-Results header field enabled by default; instead, it should be ignored, at least for the purposes of enacting filtering decisions, unless specifically enabled by the user or administrator after verifying that the border MTA is compliant. It is acceptable to have an MUA aware of this specification but have an explicit list of hostnames whose Authentication-Results header fields are trustworthy; however, this list should initially be empty.

Proposed alternative solutions to this problem were made some time ago and are listed below. To date, they have not been developed due to lack of demand but are documented here should the information be useful at some point in the future:

1. Possibly the simplest is a digital signature protecting the header field, such as using [DKIM], that can be verified by an MUA by using a posted public key. Although one of the main purposes of this document is to relieve the burden of doing message authentication work at the MUA, this only requires that

the MUA learn a single authentication scheme even if a number of them are in use at the border MTA. Note that [DKIM] requires that the From header field be signed, although in this application, the signing agent (a trusted MTA) likely cannot authenticate that value, so the fact that it is signed should be ignored. Where the authserv-id is the ADMD's domain name, the authserv-id matching this valid internal signature's "d=" DKIM value is sufficient.

2. Another would be a means to interrogate the MTA that added the header field to see if it is actually providing any message authentication services and saw the message in question, but this isn't especially palatable given the work required to craft and implement such a scheme.

3. Yet another might be a method to interrogate the internal MTAs that apparently handled the message (based on Received header fields) to determine whether any of them conform to Section 5 of this memo. This, too, has potentially high barriers to entry.

4. Extensions to [IMAP], [SMTP], and [POP3] could be defined to allow an MUA or filtering agent to acquire the authserv-id in use within an ADMD, thus allowing it to identify which Authentication-Results header fields it can trust.

5. On the presumption that internal MTAs are fully compliant with Section 3.6 of [MAIL] and the compliant internal MTAs are using their own hostnames or the ADMD's DNS domain name as the authserv-id token, the header field proposed here should always appear above a Received header added by a trusted MTA. This can be used as a test for header field validity.

Support for some of these is being considered for future work.

In any case, a mechanism needs to exist for an MUA or filter to verify that the host that appears to have added the header field (a) actually did so and (b) is legitimately adding that header field for this delivery. Given the variety of messaging environments deployed today, consensus appears to be that specifying a particular mechanism for doing so is not appropriate for this document.

Mitigation of the forged header field attack can also be accomplished by moving the authentication results data into meta-data associated with the message. In particular, an [SMTP] extension could be established to communicate authentication results from the border MTA to intermediate and delivery MTAs; the latter of these could arrange to store the authentication results as meta-data retrieved and rendered along with the message by an [IMAP] client aware of a

similar extension in that protocol. The delivery MTA would be told to trust data via this extension only from MTAs it trusts, and border MTAs would not accept data via this extension from any source. There is no vector in such an arrangement for forgery of authentication data by an outside agent.

7.2. Misleading Results

Until some form of service for querying the reputation of a sending agent is widely deployed, the existence of this header field indicating a "pass" does not render the message trustworthy. It is possible for an arriving piece of spam or other undesirable mail to pass checks by several of the methods enumerated above (e.g., a piece of spam signed using [DKIM] by the originator of the spam, which might be a spammer or a compromised system). In particular, this issue is not resolved by forged header field removal discussed above.

Hence, MUAs and downstream filters must take some care with use of this header even after possibly malicious headers are scrubbed.

7.3. Header Field Position

Despite the requirements of [MAIL], header fields can sometimes be reordered en route by intermediate MTAs. The goal of requiring header field addition only at the top of a message is an acknowledgment that some MTAs do reorder header fields, but most do not. Thus, in the general case, there will be some indication of which MTAs (if any) handled the message after the addition of the header field defined here.

7.4. Reverse IP Query Denial-of-Service Attacks

Section 4.6.4 of [SPF] describes a DNS-based denial-of-service attack for verifiers that attempt DNS-based identity verification of arriving client connections. A verifier wishing to do this check and report this information needs to take care not to go to unbounded lengths to resolve "A" and "PTR" queries. MUAs or other filters making use of an "iprev" result specified by this document need to be aware of the algorithm used by the verifier reporting the result and, especially, its limitations.

7.5. Mitigation of Backscatter

Failing to follow the instructions of Section 4.2 can result in a denial-of-service attack caused by the generation of [DSN] messages (or equivalent) to addresses that did not send the messages being rejected.

7.6. Internal MTA Lists

Section 5 describes a procedure for scrubbing header fields that may contain forged authentication results about a message. A compliant installation will have to include, at each MTA, a list of other MTAs known to be compliant and trustworthy. Failing to keep this list current as internal infrastructure changes may expose an ADMD to attack.

7.7. Attacks against Authentication Methods

If an attack becomes known against an authentication method, clearly then the agent verifying that method can be fooled into thinking an inauthentic message is authentic, and thus the value of this header field can be misleading. It follows that any attack against the authentication methods supported by this document is also a security consideration here.

7.8. Intentionally Malformed Header Fields

It is possible for an attacker to add an Authentication-Results header field that is extraordinarily large or otherwise malformed in an attempt to discover or exploit weaknesses in header field parsing code. Implementers must thoroughly verify all such header fields received from MTAs and be robust against intentionally as well as unintentionally malformed header fields.

7.9. Compromised Internal Hosts

An internal MUA or MTA that has been compromised could generate mail with a forged From header field and a forged Authentication-Results header field that endorses it. Although it is clearly a larger concern to have compromised internal machines than it is to prove the value of this header field, this risk can be mitigated by arranging that internal MTAs will remove this header field if it claims to have been added by a trusted border MTA (as described above), yet the [SMTP] connection is not coming from an internal machine known to be running an authorized MTA. However, in such a configuration, legitimate MTAs will have to add this header field when legitimate internal-only messages are generated. This is also covered in Section 5.

7.10. Encapsulated Instances

MIME messages can contain attachments of type "message/rfc822", which contain other messages. Such an encapsulated message can also contain an Authentication-Results header field. Although the processing of these is outside of the intended scope of this document

(see Section 1.3), some early guidance to MUA developers is appropriate here.

Since MTAs are unlikely to strip Authentication-Results header fields after mailbox delivery, MUAs are advised in Section 4.1 to ignore such instances within MIME attachments. Moreover, when extracting a message digest to separate mail store messages or other media, such header fields should be removed so that they will never be interpreted improperly by MUAs that might later consume them.

7.11. Reverse Mapping

Although Section 3 of this memo includes explicit support for the "iprev" method, its value as an authentication mechanism is limited. Implementers of both this proposal and agents that use the data it relays are encouraged to become familiar with the issues raised by [DNSOP-REVERSE] when deciding whether or not to include support for "iprev".

8. References

8.1. Normative References

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

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

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

[MAIL] Resnick, P., Ed., "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.

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

8.2. Informative References

[ADSP] Allman, E., Fenton, J., Delany, M., and J. Levine, "DomainKeys Identified Mail (DKIM) Author Domain Signing Practices (ADSP)", RFC 5617, August 2009.

[AR-VBR] Kucherawy, M., "Authentication-Results Registration for Vouch by Reference Results", RFC 6212, April 2011.

[ATPS] Kucherawy, M., "DomainKeys Identified Mail (DKIM) Authorized Third-Party Signatures", RFC 6541, February 2012.

[AUTH] Siemborski, R. and A. Melnikov, "SMTP Service Extension for Authentication", RFC 4954, July 2007.

[AUTH-ESC] Kucherawy, M., "Email Authentication Status Codes", RFC 7372, September 2014.

[DKIM] Crocker, D., Hansen, T., and M. Kucherawy, "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, September 2011.

[DMARC] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting and Conformance (DMARC)", RFC 7489, March 2015.

[DNS] Mockapetris, P., "Domain names - Implementation and Specification", STD 13, RFC 1035, November 1987.

[DNS-IP6] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", RFC 3596, October 2003.

[DNSOP-REVERSE] Senie, D. and A. Sullivan, "Considerations for the use of DNS Reverse Mapping", Work in Progress, March 2008.

[DOMAINKEYS] Delany, M., "Domain-Based Email Authentication Using Public Keys Advertised in the DNS (DomainKeys)", RFC 4870, May 2007.

[DSN] Moore, K. and G. Vaudreuil, "An Extensible

Message Format for Delivery Status Notifications", RFC 3464, January 2003.

[EMAIL-ARCH] Crocker, D., "Internet Mail Architecture", RFC 5598, July 2009.

[IANA-CONSIDERATIONS] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[IMAP] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[POP3] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, May 1996.

[PRA] Lyon, J., "Purported Responsible Address in E-Mail Messages", RFC 4407, April 2006.

[PTYPES-REGISTRY] Kucherawy, M., "A Property Types Registry for the Authentication-Results Header Field", RFC 7410, December 2014.

[RFC5451] Kucherawy, M., "Message Header Field for Indicating Message Authentication Status", RFC 5451, April 2009.

[RFC6008] Kucherawy, M., "Authentication-Results Registration for Differentiating among Cryptographic Results", RFC 6008, September 2010.

[RFC6577] Kucherawy, M., "Authentication-Results Registration Update for Sender Policy Framework (SPF) Results", RFC 6577, March 2012.

[RFC7001] Kucherawy, M., "Message Header Field for Indicating Message Authentication Status", RFC 7001, September 2013.

[RRVS] Mills, W. and M. Kucherawy, "The Require-Recipient-Valid-Since Header Field and SMTP Service Extension", RFC 7293, July 2014.

[SECURITY] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[SENDERID] Lyon, J. and M. Wong, "Sender ID: Authenticating E-Mail", RFC 4406, April 2006.

[SMIME-REG] Melnikov, A., "Authentication-Results Registration for S/MIME Signature Verification", RFC 7281, June 2014.

[SPF] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1", RFC 7208, April 2014.

[VBR] Hoffman, P., Levine, J., and A. Hathcock, "Vouch By Reference", RFC 5518, April 2009.

Appendix A. Acknowledgments

The author wishes to acknowledge the following individuals for their review and constructive criticism of this document: Stephane Bortzmeyer, Scott Kitterman, John Levine, Tom Petch, and Pete Resnick.

Appendix B. Legacy MUAs

Implementers of this protocol should be aware that many MUAs are unlikely to be retrofitted to support the new header field and its semantics. In the interests of convenience and quicker adoption, a delivery MTA might want to consider adding things that are processed by existing MUAs in addition to the Authentication-Results header field. One suggestion is to include a Priority header field, on messages that don't already have such a header field, containing a value that reflects the strength of the authentication that was accomplished, e.g., "low" for weak or no authentication, "normal" or "high" for good or strong authentication.

Some modern MUAs can already filter based on the content of this header field. However, there is keen interest in having MUAs make some kind of graphical representation of this header field's meaning to end users. Until this capability is added, other interim means of conveying authentication results may be necessary while this proposal and its successors are adopted.

Appendix C. Authentication-Results Examples

This section presents some examples of the use of this header field to indicate authentication results.

C.1. Trivial Case; Header Field Not Present

The trivial case:

```
Received: from mail-router.example.com
              (mail-router.example.com [192.0.2.1])
          by server.example.org (8.11.6/8.11.6)
              with ESMTP id g1G0r1kA003489;
          Fri, Feb 15 2002 17:19:07 -0800
From: sender@example.com
Date: Fri, Feb 15 2002 16:54:30 -0800
To: receiver@example.org
Message-Id: <12345.abc@example.com>
Subject: here's a sample

Hello!  Goodbye!
```

Example 1: Trivial Case

The Authentication-Results header field is completely absent. The MUA may make no conclusion about the validity of the message. This could be the case because the message authentication services were not available at the time of delivery, or no service is provided, or the MTA is not in compliance with this specification.

C.2. Nearly Trivial Case; Service Provided, but No Authentication Done

A message that was delivered by an MTA that conforms to this specification but provides no actual message authentication service:

```
Authentication-Results: example.org 1; none
Received: from mail-router.example.com
              (mail-router.example.com [192.0.2.1])
          by server.example.org (8.11.6/8.11.6)
              with ESMTP id g1G0r1kA003489;
          Fri, Feb 15 2002 17:19:07 -0800
From: sender@example.com
Date: Fri, Feb 15 2002 16:54:30 -0800
To: receiver@example.org
Message-Id: <12345.abc@example.com>
Subject: here's a sample

Hello!  Goodbye!
```

Example 2: Header Present but No Authentication Done

The Authentication-Results header field is present, showing that the delivering MTA conforms to this specification. It used its DNS

domain name as the authserv-id. The presence of "none" (and the absence of any method and result tokens) indicates that no message authentication was done. The version number of the specification to which the field's content conforms is explicitly provided.

C.3. Service Provided, Authentication Done

A message that was delivered by an MTA that conforms to this specification and applied some message authentication:

```
Authentication-Results: example.com;
          spf=pass smtp.mailfrom=example.net
Received: from dialup-1-2-3-4.example.net
              (dialup-1-2-3-4.example.net [192.0.2.200])
          by mail-router.example.com (8.11.6/8.11.6)
              with ESMTP id g1G0r1kA003489;
          Fri, Feb 15 2002 17:19:07 -0800
From: sender@example.net
Date: Fri, Feb 15 2002 16:54:30 -0800
To: receiver@example.com
Message-Id: <12345.abc@example.net>
Subject: here's a sample

Hello!  Goodbye!
```

Example 3: Header Reporting Results

The Authentication-Results header field is present, indicating that the border MTA conforms to this specification. The authserv-id is once again the DNS domain name. Furthermore, the message was authenticated by that MTA via the method specified in [SPF]. Note that since that method cannot authenticate the local-part, it has been omitted from the result's value. The MUA could extract and relay this extra information if desired.

C.4. Service Provided, Several Authentications Done, Single MTA

A message that was relayed inbound via a single MTA that conforms to this specification and applied three different message authentication checks:

```
Authentication-Results: example.com;
          auth=pass (cram-md5) smtp.auth=sender@example.net;
          spf=pass smtp.mailfrom=example.net
Authentication-Results: example.com;
          sender-id=pass header.from=example.net
Received: from dialup-1-2-3-4.example.net (8.11.6/8.11.6)
              (dialup-1-2-3-4.example.net [192.0.2.200])
          by mail-router.example.com (8.11.6/8.11.6)
              with ESMTPA id g1G0r1kA003489;
          Fri, Feb 15 2002 17:19:07 -0800
Date: Fri, Feb 15 2002 16:54:30 -0800
To: receiver@example.com
From: sender@example.net
Message-Id: <12345.abc@example.net>
Subject: here's a sample

Hello!  Goodbye!
```

Example 4: Headers Reporting Results from One MTA

The Authentication-Results header field is present, indicating that the delivering MTA conforms to this specification. Once again, the receiving DNS domain name is used as the authserv-id. Furthermore, the sender authenticated herself/himself to the MTA via a method specified in [AUTH], and both SPF and Sender ID checks were done and passed. The MUA could extract and relay this extra information if desired.

Two Authentication-Results header fields are not required since the same host did all of the checking. The authenticating agent could have consolidated all the results into one header field.

This example illustrates a scenario in which a remote user on a dialup connection (example.net) sends mail to a border MTA (example.com) using SMTP authentication to prove identity. The dialup provider has been explicitly authorized to relay mail as example.com resulting in passes by the SPF and Sender ID checks.

C.5. Service Provided, Several Authentications Done, Different MTAs

A message that was relayed inbound by two different MTAs that conform to this specification and applied multiple message authentication checks:

```
Authentication-Results: example.com;
          sender-id=fail header.from=example.com;
          dkim=pass (good signature) header.d=example.com
Received: from mail-router.example.com
              (mail-router.example.com [192.0.2.1])
          by auth-checker.example.com (8.11.6/8.11.6)
              with ESMTP id i7PK0sH7021929;
          Fri, Feb 15 2002 17:19:22 -0800
DKIM-Signature:  v=1; a=rsa-sha256; s=gatsby; d=example.com;
          t=1188964191; c=simple/simple; h=From:Date:To:Subject:
          Message-Id:Authentication-Results;
          bh=sEuZGD/pSr7ANysbY3jtdaQ3Xv9xPQtS0m70;
          b=EToRSuvUfQVP3Bkz ... rTB0t0gYnBVCM=
Authentication-Results: example.com;
          auth=pass (cram-md5) smtp.auth=sender@example.com;
          spf=fail smtp.mailfrom=example.com
Received: from dialup-1-2-3-4.example.net
              (dialup-1-2-3-4.example.net [192.0.2.200])
          by mail-router.example.com (8.11.6/8.11.6)
              with ESMTPA id g1G0r1kA003489;
          Fri, Feb 15 2002 17:19:07 -0800
From: sender@example.com
Date: Fri, Feb 15 2002 16:54:30 -0800
To: receiver@example.com
Message-Id: <12345.abc@example.com>
Subject: here's a sample

Hello!  Goodbye!
```

Example 5: Headers Reporting Results from Multiple MTAs

The Authentication-Results header field is present, indicating conformance to this specification. Once again, the authserv-id used is the recipient's DNS domain name. The header field is present twice because two different MTAs in the chain of delivery did authentication tests. The first MTA, mail-router.example.com, reports that SMTP AUTH and SPF were both used and that the former passed while the latter failed. In the SMTP AUTH case, additional information is provided in the comment field, which the MUA can choose to render if desired.

The second MTA, auth-checker.example.com, reports that it did a

Sender ID test (which failed) and a DKIM test (which passed). Again, additional data about one of the tests is provided as a comment, which the MUA may choose to render. Also noteworthy here is the fact that there is a DKIM signature added by example.com that assured the integrity of the lower Authentication-Results field.

Since different hosts did the two sets of authentication checks, the header fields cannot be consolidated in this example.

This example illustrates more typical transmission of mail into example.com from a user on a dialup connection example.net. The user appears to be legitimate as he/she had a valid password allowing authentication at the border MTA using SMTP AUTH. The SPF and Sender ID tests failed since example.com has not granted example.net authority to relay mail on its behalf. However, the DKIM test passed because the sending user had a private key matching one of example.com's published public keys and used it to sign the message.

C.6. Service Provided, Multi-Tiered Authentication Done

A message that had authentication done at various stages, one of which was outside the receiving ADMD:

```
Authentication-Results: example.com;
      dkim=pass reason="good signature"
        header.i=@mail-router.example.net;
      dkim=fail reason="bad signature"
        header.i=@newyork.example.com
Received: from mail-router.example.net
          (mail-router.example.net [192.0.2.250])
      by chicago.example.com (8.11.6/8.11.6)
          for <recipient@chicago.example.com>
          with ESMTP id i7PK0sH7021929;
      Fri, Feb 15 2002 17:19:22 -0800
DKIM-Signature: v=1; a=rsa-sha256; s=furble;
      d=mail-router.example.net; t=1188964198; c=relaxed/simple;
      h=From:Date:To:Message-Id:Subject:Authentication-Results;
      bh=ftA9J6GtX8OpwUECzHnCkRzKw1uk6FNiLfJl5Nmv49E=;
      b=oINEO8hgn/gnunsg ... 9n9ODSNFSDij3=
Authentication-Results: example.net;
      dkim=pass (good signature) header.i=@newyork.example.com
Received: from smtp.newyork.example.com
          (smtp.newyork.example.com [192.0.2.220])
      by mail-router.example.net (8.11.6/8.11.6)
          with ESMTP id g1G0r1kA003489;
      Fri, Feb 15 2002 17:19:07 -0800
DKIM-Signature: v=1; a=rsa-sha256; s=gatsby;
      d=newyork.example.com;
      t=1188964191; c=simple/simple;
      h=From:Date:To:Message-Id:Subject;
      bh=sEu28nfs9fuZGD/pSr7ANysbY3jtdaQ3Xv9xPQtS0m7=;
      b=EToRSuvUfQVP3Bkz ... rTB0t0gYnBVCM=
From: sender@newyork.example.com
Date: Fri, Feb 15 2002 16:54:30 -0800
To: meetings@example.net
Message-Id: <12345.abc@newyork.example.com>
Subject: here's a sample
```

Example 6: Headers Reporting Results from Multiple MTAs in Different ADMDs

In this example, we see multi-tiered authentication with an extended trust boundary.

The message was sent from someone at example.com's New York office (newyork.example.com) to a mailing list managed at an intermediary.

The message was signed at the origin using DKIM.

The message was sent to a mailing list service provider called example.net, which is used by example.com. There, meetings@example.net is expanded to a long list of recipients, one of whom is at the Chicago office. In this example, we will assume that the trust boundary for chicago.example.com includes the mailing list server at example.net.

The mailing list server there first authenticated the message and affixed an Authentication-Results header field indicating such using its DNS domain name for the authserv-id. It then altered the message by affixing some footer text to the body, including some administrivia such as unsubscription instructions. Finally, the mailing list server affixes a second DKIM signature and begins distribution of the message.

The border MTA for chicago.example.com explicitly trusts results from mail-router.example.net, so that header field is not removed. It performs evaluation of both signatures and determines that the first (most recent) is a "pass" but, because of the aforementioned modifications, the second is a "fail". However, the first signature included the Authentication-Results header added at mail-router.example.net that validated the second signature. Thus, indirectly, it can be determined that the authentications claimed by both signatures are indeed valid.

Note that two styles of presenting meta-data about the result are in use here. In one case, the "reason=" clause is present, which is intended for easy extraction by parsers; in the other case, the CFWS production of the ABNF is used to include such data as a header field comment. The latter can be harder for parsers to extract given the varied supported syntaxes of mail header fields.

C.7. Comment-Heavy Example

The formal syntax permits comments within the content in a number of places. For the sake of illustration, this example is also legal:

```
Authentication-Results: foo.example.net (foobar) 1 (baz);
    dkim (Because I like it) / 1 (One yay) = (wait for it) fail
      policy (A dot can go here) . (like that) expired
      (this surprised me) = (as I wasn't expecting it) 1362471462
```

Example 7: A Very Comment-Heavy but Perfectly Legal Example

Appendix D. Operational Considerations about Message Authentication

This protocol is predicated on the idea that authentication (and presumably in the future, reputation) work is typically done by border MTAs rather than MUAs or intermediate MTAs; the latter merely make use of the results determined by the former. Certainly this is not mandatory for participation in electronic mail or message authentication, but this protocol and its deployment to date are based on that model. The assumption satisfies several common ADMD requirements:

1. Service operators prefer to resolve the handling of problem messages as close to the border of the ADMD as possible. This enables, for example, rejection of messages at the SMTP level rather than generating a DSN internally. Thus, doing any of the authentication or reputation work exclusively at the MUA or intermediate MTA renders this desire unattainable.

2. Border MTAs are more likely to have direct access to external sources of authentication or reputation information since modern MUAs are more likely to be heavily firewalled. Thus, some MUAs might not even be able to complete the task of performing authentication or reputation evaluations without complex proxy configurations or similar burdens.

3. MUAs rely upon the upstream MTAs within their trust boundaries to make correct (as much as is possible) evaluations about the message's envelope, header, and content. Thus, MUAs don't need to know how to do the work that upstream MTAs do; they only need the results of that work.

4. Evaluations about the quality of a message, from simple token matching (e.g., a list of preferred DNS domains) to cryptanalysis (e.g., public/private key work), are at least a little bit expensive and thus need to be minimized. To that end, performing those tests at the border MTA is far preferred to doing that work at each MUA that handles a message. If an ADMD's environment adheres to common messaging protocols, a reputation query or an authentication check performed by a border MTA would return the same result as the same query performed by an MUA. By contrast, in an environment where the MUA does the work, a message arriving for multiple recipients would thus cause authentication or reputation evaluation to be done more than once for the same message (i.e., at each MUA), causing needless amplification of resource use and creating a possible denial-of-service attack vector.

5. Minimizing change is good. As new authentication and reputation methods emerge, the list of methods supported by this header field would presumably be extended. If MUAs simply consume the contents of this header field rather than actually attempt to do authentication and/or reputation work, then MUAs only need to learn to parse this header field once; emergence of new methods requires only a configuration change at the MUAs and software changes at the MTAs (which are presumably fewer in number). When choosing to implement these functions in MTAs vs. MUAs, the issues of individual flexibility, infrastructure inertia, and scale of effort must be considered. It is typically easier to change a single MUA than an MTA because the modification affects fewer users and can be pursued with less care. However, changing many MUAs is more effort than changing a smaller number of MTAs.

6. For decisions affecting message delivery and display, assessment based on authentication and reputation is best performed close to the time of message transit, as a message makes its journey toward a user's inbox, not afterwards. DKIM keys and IP address reputations, etc., can change over time or even become invalid, and users can take a long time to read a message once delivered. The value of this work thus degrades, perhaps quickly, once the delivery process has completed. This seriously diminishes the value of this work when done other than at MTAs.

Many operational choices are possible within an ADMD, including the venue for performing authentication and/or reputation assessment. The current specification does not dictate any of those choices. Rather, it facilitates those cases in which information produced by one stage of analysis needs to be transported with the message to the next stage.

Appendix E. Change Since RFC7001

- Apply RFC7410.
- Update all the RFC4408 references to RFC7208.
- Add section explaining "property" values. (Errata #4201)
- Some minor text reorganization.
- Give registry history, enough that this is now the authoritative registry definition.
- Add text explaining each of the method-ptype-property tuples registered by this document.

- Change the meaning of the "Defined" column of the methods registry to be the place where each entry was created and described; it is expected that this will then refer to the method's defining document. Provide IANA with corresponding update instructions.
- Clean up registry structure and content, and replace all RFC7001 references with pointers to this document.
- Add references: [DMARC], [PRA], [RFC6008], [RFC6577], [RRVS], [SMIME-REG].
- Add description of values that can be extracted from SMTP AUTH sessions and an example.
- Much more complete descriptions of reporting DomainKeys results.
- Add more detail about Sender ID.
- Mark all ADSP and DomainKeys entries as deprecated since their defining documents are as well.
- Rework some text around ignoring unknown ptypes.
- Completely describe the ptypes registry.
- EHLO is mapped to HELO for SPF.
- RFC7208 uses all-lowercase result strings now, so adjust prose accordingly.
- Update list of supported methods, and mention the registries immediately below there.
- Mention that when a local-part is removed, the "@" goes with it.
- Refer to RFC7328 in the "iprev" definition.
- Correction to "smime-part" prose.
- Examples that use SMTP AUTH now claim "with ESMTPA" in the Received fields.
- Minor editorial adjustments.

Author's Address

Murray S. Kucherawy
270 Upland Drive
San Francisco, CA 94127
US

EMail: superuser@gmail.com

Applications Area Working Group
Internet-Draft
Intended Status: Informational
Expires: April 19, 2016

S. Leonard
Penango, Inc.

October 17, 2015

The text/markdown Media Type
draft-ietf-appsawg-text-markdown-12

Abstract

This document registers the text/markdown media type for use with Markdown, a family of plain text formatting syntaxes that optionally can be converted to formal markup languages such as HTML.

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."

Copyright Notice

Table of Contents

1. Introduction

1.1. This Is Markdown! Or: Markup and Its Discontents

In computer systems, textual data is stored and processed using a continuum of techniques. On the one end is plain text: computer-encoded text that consists only of a sequence of code points from a given standard, with no other formatting or structural information [UNICODE]. (On the other end is binary data, which computer systems store and process with bit-for-bit accuracy.) Many of these standards include control characters that are used as in-band signaling to cause effects other than the addition of a symbol (or grapheme) to the text.

Markup offers an alternative means to encode this signaling information by overloading certain graphic characters (see, e.g., [ISO646]) with additional meanings. Therefore, markup languages allow for annotating a document in a syntactically distinguishable way from the text, while keeping the annotations printable. Markup languages are (reasonably) well-specified and tend to follow (mostly) standardized syntax rules. Examples of formal markup languages include SGML, HTML, XML, and LaTeX. Standardized rules lead to interoperability between markup processors, but impose skill requirements on new users that lead to markup languages becoming less accessible to beginners. These rules also reify "validity": content that does not conform to the rules is treated differently (i.e., is rejected) than content that conforms.

In contrast to formal markup languages, lightweight markup languages use simple syntaxes; they are designed to be easy for humans to enter and understand with basic text editors. Markdown, the subject of this document, began as an /informal/ plain text formatting syntax [MDSYNTAX] and Perl script HTML/XHTML processor [MARKDOWN] targeted at non-technical users using unspecialized tools, such as plain text e-mail clients. [MDSYNTAX] explicitly rejects the notion of validity: there is no such thing as "invalid" Markdown. If the Markdown content does not result in the "right" output (defined as output that the author wants, not output that adheres to some dictated system of rules), the expectation is that the author should continue experimenting by changing the content or the processor to achieve the desired output.

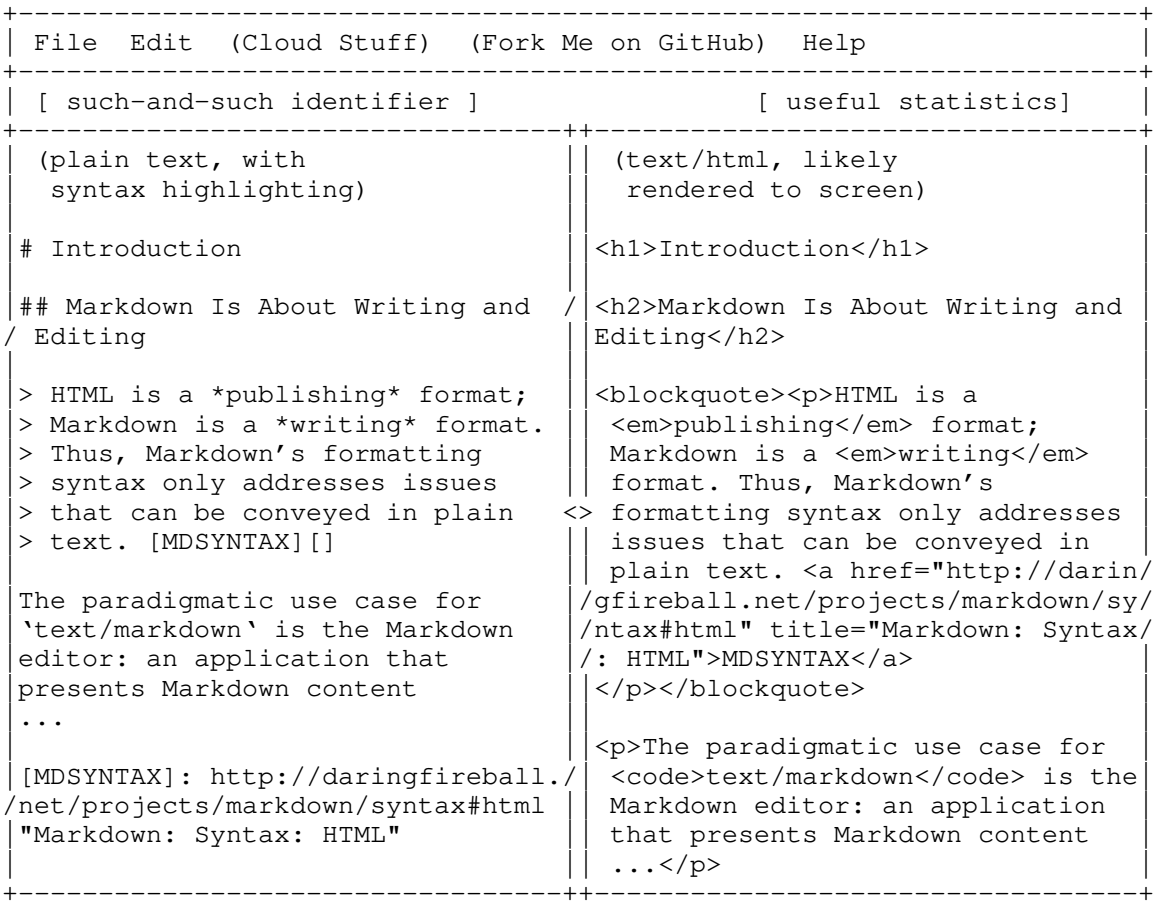
Since its development in 2004 [MARKDOWN], a number of web- and Internet-facing applications have incorporated Markdown into their text entry systems, frequently with custom extensions. Markdown has thus evolved into a kind of Internet meme [INETMEME] as different communities encounter it and adapt the syntax for their specific use cases. Markdown now represents a family of related plain text formatting syntaxes and implementations that, while broadly compatible with humans [HUMANE], are intended to produce different kinds of outputs that push the boundaries of mutual intelligibility between software systems.

To support identifying and conveying Markdown, this document defines a media type and parameters that indicate the Markdown author's intent on how to interpret the content. This registration draws particular inspiration from text/troff [RFC4263], which is a plain text formatting syntax for typesetting based on tools from the 1960s ("RUNOFF") and 1970s ("nroff", et. al.). In that sense, Markdown is a kind of troff for modern computing. A companion document [MDMTGUID] provides additional Markdown background, philosophy, local storage strategies, and variant registrations (including examples).

1.2. Markdown Is About Writing and Editing

> "HTML is a *publishing* format; Markdown is a *writing* format. Thus, Markdown's formatting syntax only addresses issues that can be conveyed in plain text." [MDSYNTAX]

The paradigmatic use case for text/markdown is the Markdown editor: an application that presents Markdown content (which looks like an e-mail or other piece of plain text writing) alongside a published format, so that an author can see results instantaneously and can tweak his or her input in real-time. A significant number of Markdown editors have adopted "split-screen view" (or "live preview") technology that looks like Figure 1:



LEGEND: "/" embedded in a vertical line represents a line-continuation marker, since a line break is not supposed to occur in that content.

Figure 1: Markdown Split-Screen/Live Preview Editor

To get the best results, implementations ought to produce and consume mutually intelligible and identifiable bits of Markdown. That way, users on diverse platforms can collaborate with their tools of choice. Those tools can be desktop-based (MarkdownPad, MultiMarkdown Composer), browser-based (Dillinger, Markable), integrated widgets (Discourse, GitHub), general-purpose editors (emacs, vi), or plain old "Notepad". Additionally, implementations ought to have common ways to identify particular areas of Markdown content when the Markdown becomes appreciably large (e.g., book chapters and Internet-Drafts--not just blog posts). So that users have the option to use Markdown in MIME-capable systems to convey their works in progress, not just their

finished products (for which full-blown markups ranging from text/html to application/pdf are appropriate), implementations ought to label such Markdown content with a common media type: text/markdown. This registration facilitates interoperability between these Markdown editors by conveying the syntax of the particular Markdown variant and the desired output format.

1.3. Definitions

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].

Since Markdown signifies a family of related formats with varying degrees of formal documentation and implementation, this specification uses the term "variant" to identify such formats.

2. Markdown Media Type Registration Application

This section provides the media type registration application for the text/markdown media type (see [RFC6838], Section 5.6).

Type name: text

Subtype name: markdown

Required parameters:

charset: Per Section 4.2.1 of [RFC6838], charset is REQUIRED. There is no default value because neither [MDSYNTAX] nor many popular implementations at the time of this registration do either. [MDSYNTAX] clearly describes Markdown as a "writing format"; its syntax rules operate on characters (specifically, on punctuation) rather than code points. Many Markdown processors will get along just fine by operating on characters in the US-ASCII repertoire (specifically punctuation), blissfully oblivious to other characters or codes.

Optional parameters:

variant: An optional identifier of the specific Markdown variant that the author intended. The value serves as a "hint" to the recipient, meaning that the recipient MAY interpret the Markdown as that variant, but is under no obligation to do so. When omitted, there is no hint; the interpretation is entirely up to the receiver and context. This identifier is plain US-ASCII and case-insensitive. To promote interoperability, identifiers can be registered in the registry defined in Section 6. If a receiver

does not recognize the variant identifier, the receiver MAY present the identifier to a user to inform him or her of it.

Other parameters MAY be included with the media type. The variant SHOULD define the semantics of such other parameters. Additionally, the variant MAY be registered under another media type; this text/markdown registration does not preclude other registrations.

Encoding considerations:

Markdown content is plain text content; any octet sequence is valid as long as it conforms to the character codes of the charset parameter. See [RFC2046]. Markup characters in [MDSYNTAX] are limited to printable US-ASCII; however, other variants can define markup characters outside this range (including control characters such as NUL and characters encoded in multiple octets).

Security considerations:

Markdown interpreted as plain text is relatively harmless. A text editor need only display the text. The editor SHOULD take care to handle control characters appropriately, and to limit the effect of the Markdown to the text editing area itself; malicious Unicode-based Markdown could, for example, surreptitiously change the directionality of the text. An editor for normal text would already take these control characters into consideration, however.

Markdown interpreted as a precursor to other formats, such as HTML, carries all of the security considerations as the target formats. For example, HTML can contain instructions to execute scripts, redirect the user to other webpages, download remote content, and upload personally identifiable information. Markdown also can contain islands of formal markup, such as HTML. These islands of formal markup may be passed as-is, transformed, or ignored (perhaps because the islands are conditional or incompatible) when the Markdown is processed. Since Markdown may have different interpretations depending on the tool and the environment, a better approach is to analyze (and sanitize or block) the output markup, rather than attempting to analyze the Markdown.

Interoperability considerations:

Markdown variations (some might say "innovations") are designed to be broadly compatible with humans ("humane"), but not necessarily with each other. Therefore, syntax in one Markdown derivative may be ignored or treated differently in another derivative. The overall effect is a general degradation of the output that increases with the quantity of variant-specific Markdown used in

the text. When it is desirable to reflect the author's intent in the output, stick with the variant identified in the variant parameter, i.e., receivers SHOULD only accept Markdown variants that they explicitly know about, and senders SHOULD avoid use of variants that intended recipients are not known to understand.

Published specification: This specification; [MDSYNTAX].

Applications that use this media type:

Markdown conversion tools, Markdown WYSIWYG editors, and plain text editors and viewers; markup processor targets indirectly use Markdown (e.g., web browsers for Markdown converted to HTML).

Fragment identifier considerations:

See Section 3.

Additional information:

Magic number(s): None
File extension(s): .md, .markdown
Macintosh file type code(s):
TEXT. A uniform type identifier (UTI) of "net.daringfireball.markdown", which conforms to "public.plain-text", is RECOMMENDED [MDUTI]. See [MDMTGUID] for other considerations.

Person & email address to contact for further information:

Sean Leonard <dev+ietf@seantek.com>

Restrictions on usage: None.

Author/Change controller: Sean Leonard <dev+ietf@seantek.com>

Intended usage: COMMON

Provisional registration? No

Implementations SHOULD record the value of the variant parameter (and other parameters if defined by the variant) along with the Markdown content when the content leaves the domain of Internet media type-capable formats. Strategies for doing so are discussed in [MDMTGUID].

The Content-Disposition header (particularly the preview-type parameter) can be used with Markdown content. See Section 4.

3. Fragment Identifiers

[MARKDOWN] does not define any fragment identifiers, but some variants do, and many types of Markdown processor output (e.g., HTML or PDF) will have well-defined fragment identifiers. Which fragment identifiers are available for a given document are variant-defined.

When encoded in a URI, characters that are outside of the fragment production of [RFC3986] are percent-encoded. The default encoding (character set) of percent-encoded octets in URIs is the same as the Markdown content, which is identified by the charset parameter or by other contextual means. Fragment identifiers SHOULD be considered case-sensitive, which maintains consistency with HTML. Variants MAY override the guidance in this paragraph.

At least the first equals sign "=" SHOULD be percent-encoded to prevent ambiguity as described in the following section.

3.1. Parameters

Similar to application/pdf [RFC3778] and text/plain [RFC5147], this registration permits a parameter syntax for fragment identifiers. The syntax is a parameter name, the equals sign "=" (which MUST NOT be percent-encoded), and a parameter value. To the extent that multiple parameters can appear in a fragment production, the parameters SHALL be separated by the ampersand "&" (which MUST NOT be percent-encoded).

The only parameter defined in this registration is "line", which has the same meaning as [RFC5147] (i.e., counting is zero-based). For example: "#line=10" identifies the eleventh line of Markdown input. Implementers should take heed that different environments and character sets may have a wide range of code sequences to divide lines.

Markdown variants are free to define additional parameters.

4. Content Disposition and preview-type

The Content-Disposition header [RFC2183] conveys presentational information about a MIME entity, using a type and set of parameters. The parameter "preview-type" is defined here for Markdown content.

When present, "preview-type" indicates the Internet media type (and parameters) of the preview output desired from the processor by the author. With reference to the "paradigmatic use case" (i.e., collaborative Markdown editing) in Section 1.3, the preview-type parameter primarily affects the "right-hand" side of a Markdown

editor. There is no default value: when absent, a Markdown user agent can render or display whatever it wants.

The value of this parameter is an Internet media type with optional parameters. The syntax (including case sensitivity considerations) is the same as specified in [RFC2045] for the Content-Type header (with updates over time, e.g., [RFC2231] and [RFC6532]).

Implementations SHOULD anticipate and support HTML (text/html) and XHTML (application/xhtml+xml) output, to the extent that a syntax targets those markup languages. These types ought to be suitable for the majority of current purposes. However, Markdown is increasingly becoming integral to workflows where HTML is not the target output; examples range from TeX, to PDF, to OPML, and even to entire e-books (e.g., [PANDOC]).

The reflexive media type "text/markdown" in this parameter value means that the author does not want to invoke Markdown processing at all: the receiver SHOULD present the Markdown source as-is.

The "preview-type" parameter can be used for other types of content, but the precise semantics are not defined here.

5. Example

The following is an example of Markdown as an e-mail attachment:

```
MIME-Version: 1.0
Content-Type: text/markdown; charset=UTF-8; variant=Original
Content-Disposition: attachment; filename=readme.md;
 preview-type="application/xhtml+xml"

Sample HTML 4 Markdown
==========

This is some sample Markdown. [Hooray!][foo]
(Remember that link identifiers are not case-sensitive.)

Bulleted Lists
-------

Here are some bulleted lists...

* One Potato
* Two Potato
* Three Potato
```

```
- One Tomato
- Two Tomato
- Three Tomato

More Information
-----------

[.markdown, .md](http://daringfireball.net/projects/markdown/)
has more information.

[fOo]: http://example.com/loc 'Will Not Work with Markdown.pl-1.0.1'
```

6. IANA Considerations

IANA is asked to register the media type text/markdown using the application provided in Section 2 of this document.

IANA is asked to register "preview-type" in the Content Disposition Parameters subregistry of the Content Disposition Values and Parameters registry, with the following description: "Internet media type (and parameters) of the preview output desired from a processor by the author of the MIME content".

6.1. Markdown Variants

IANA is also asked to establish a registry called "Markdown Variants". While the registry is being created in the context of the text/markdown media type, the registry is intended for broad community use, so protocols and systems that do not rely on Internet media types can still tag Markdown content with a common variant identifier. Each entry in this registry shall consist of basic information about the variant:

Identifier:	unique identifier for the variant
Name:	the commonly known name of the variant
Description:	a prose description of the variant, including how it differs from other variants such as Original
Additional Parameters*:	additional Content-Type parameters
Fragment Identifiers*:	additional fragment identifier syntaxes and semantics
References:	URIs or other references to documentation
Contact Information:	whom to contact (email, URI, phone, address, etc.)
Expiration Date^:	when this provisional registration expires

* (optional)

^ (if provisional)

While the variant parameter is "plain US-ASCII" (see registration template), the Identifier field (and by implication, all registered identifiers) SHALL conform to the ABNF [RFC5234]:

```
ALPHA [*VCHAR (ALPHA / DIGIT)]
```

For style and compatibility reasons, the Identifier field SHOULD conform to the ABNF:

```
ALPHA *( ["-" / "." / "_" / "~"] 1*(ALPHA / DIGIT) )
```

I.e., the identifier MUST start with a letter and MAY contain punctuation in the middle, but not at the end: the last character MUST be alphanumeric. The second production uses the same characters as the "unreserved" rule of [RFC3986], and is designed to be compatible with characters in other identification systems, e.g., filenames. Since the identifier MAY be displayed to a user--particularly in cases where the receiver does not recognize the identifier--the identifier SHOULD be rationally related to the vernacular name of the variant.

The Name, Description, Additional Parameters, Fragment Identifiers, References, and Contact Information fields SHALL be in a Unicode character set (e.g., UTF-8).

The registry includes the registration in Section 6.1.4 (Original Markdown). [MDMTGUID] includes additional registrations.

6.1.1. Reserved Identifiers

The registry has the following identifiers RESERVED, as they have engendered some controversy in the Markdown community. No one is allowed to register them (or any case variations of them). These identifiers are not and cannot be registered:

Standard
Common
Markdown

The registry includes the following text in the note field:

> The variant names Standard, Common, and Markdown are reserved and cannot be registered.

6.1.2. Standard of Review

Registrations are made on a First-Come, First-Served [RFC5226] basis by anyone with a need to interoperate. While documentation is

required, any level of documentation is sufficient; thus, neither Specification Required nor Expert Review are warranted. The checks prescribed by this section can be performed automatically.

All references (including contact information) MUST be verified as functional at the time of the registration.

As a special "escape valve", registrations can be updated with IETF Review [RFC5226]. All fields may be updated except the variant identifier, which is permanent: not even case may be changed.

6.1.3. Provisional Registration

Any registrant may make a provisional registration to reserve a variant identifier. Only the variant identifier and contact information fields are required; the rest are optional. Provisional registrations expire after three months, after which time the variant identifier may be reused. To make a registration permanent, a registrant simply needs to complete a permanent registration with the same identifier as the provisional registration.

6.1.4. Original Markdown

The registry includes this initial variant. A conforming implementation that processes the variant parameter MUST recognize this variant (although the processing behavior is not defined here).

Identifier: Original

Name: Markdown

Description:
Gruber’s original Markdown syntax.

References:
[MARKDOWN]
[MDSYNTAX]

Contact Information:
(individual) John Gruber <http://daringfireball.net/>
<comments@daringfireball.net>

7. Security Considerations

See the Security considerations entry in Section 2.

8. References

8.1. Normative References

[MARKDOWN] Gruber, J., "Daring Fireball: Markdown", December 2004, <http://daringfireball.net/projects/markdown/>.

[MDSYNTAX] Gruber, J., "Daring Fireball: Markdown Syntax Documentation", December 2004, <http://daringfireball.net/projects/markdown/syntax>.

[MDUTI] Gruber, J., "Daring Fireball: Uniform Type Identifier for Markdown", August 2011, <http://daringfireball.net/linked/2011/08/05/markdown-uti>.

[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.

[RFC2183] Troost, R., Dorner, S., and K. Moore, Ed., "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, August 1997.

[RFC2231] Freed, N. and K. Moore, "MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations", RFC 2231, November 1997.

[RFC3778] Taft, E., Pravetz, J., Zilles, S., and L. Masinter, "The application/pdf Media Type", RFC 3778, May 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5147] Wilde, E. and M. Duerst, "URI Fragment Identifiers for the text/plain Media Type", RFC 5147, April 2008.

[RFC5226] Narten, T., and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, May 2008.

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

[RFC6532] Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, February 2012.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

8.2. Informative References

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[UNICODE] The Unicode Consortium, "The Unicode Standard, Version 8.0.0", The Unicode Consortium, August 2015.

[ISO646] International Organization for Standardization, "Information technology - ISO 7-bit coded character set for information interchange", ISO Standard 646, 1991.

[HUMANE] Atwood, J., "Is HTML a Humane Markup Language?", May 2008, <http://blog.codinghorror.com/is-html-a-humane-markup-language/>.

[INETMEME] Solon, O., "Richard Dawkins on the internet's hijacking of the word 'meme'", June 2013, <http://www.wired.co.uk/news/archive/2013-06/20/richard-dawkins-memes>, <http://www.webcitation.org/6HzDGE9Go>.

[MDMTGUID] Leonard, S., "Guidance on Markdown: Design Philosophies, Stability Strategies, and Select Registrations", draft-ietf-appsawg-text-markdown-use-cases-07 (work in progress), September 2015.

[PANDOC] MacFarlane, J., "Pandoc", 2014, <http://johnmacfarlane.net/pandoc/>.

[RFC4263] Lilly, B., "Media Subtype Registration for Media Type text/troff", RFC 4263, January 2006.

Author's Address

Sean Leonard
Penango, Inc.
5900 Wilshire Boulevard
21st Floor
Los Angeles, CA 90036
USA

EMail: dev+ietf@seantek.com
URI: http://www.penango.com/

Applications Area Working Group
Internet-Draft
Intended Status: Informational
Expires: March 7, 2016

S. Leonard
Penango, Inc.

September 4, 2015

Guidance on Markdown: Design Philosophies, Stability Strategies, and Select Registrations

draft-ietf-appsawg-text-markdown-use-cases-07

Abstract

This document elaborates upon the text/markdown media type for use with Markdown, a family of plain text formatting syntaxes that optionally can be converted to formal markup languages such as HTML. Background information, local storage strategies, and additional syntax registrations are supplied.

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."

Table of Contents

1. Dive Into Markdown

This document serves as an informational companion to [MDMTREG], the text/markdown media type registration. It should be considered jointly with [MDMTREG].

> "Sometimes the truth of a thing is not so much in the
> think of it, but in the feel of it." --Stanley Kubrick

1.1. On Formats

In computer systems, textual data is stored and processed using a continuum of techniques. On the one end is plain text: computer-encoded text that consists only of a sequence of code points from a given standard, with no other formatting or structural information [UNICODE]. Plain text provides /some/ fixed facilities for formatting instructions, namely codes in the character set that have meanings other than "represent this character graphically on the output medium"; however, these facilities are not particularly extensible. Compare with [RFC6838] Section 4.2.1. Applications may neuter the effects of these special characters by prohibiting them or by ignoring their dictated meanings, as is the case with how modern applications treat most control characters in US-ASCII. On this end, any text reader or editor that interprets the character set can be used to see or manipulate the text. If some characters are corrupted, the corruption is unlikely to affect the ability of a computer system to process the text (even if the human meaning is changed).

On the other end is binary data: a sequence of bits intended for some computer application to interpret and act upon. Binary formats are flexible in that they can store non-textual data efficiently (perhaps storing no text at all, or only storing certain kinds of text for very specialized purposes). Binary formats require an application to be coded specifically to handle the format; no partial interoperability is possible. Furthermore, if even one bit is corrupted in a binary format, it may prevent an application from processing any of the data correctly.

Between these two extremes lies formatted text, i.e., text that includes non-textual information coded in a particular way, that affects the interpretation of the text by computer programs. Formatted text is distinct from plain text and binary data in that the non-textual information is encoded into textual characters, which are assigned specialized meanings not defined by the character set. With a regular text editor and a standard keyboard (or other standard input mechanism), a user can enter these textual characters to express the non-textual meanings. For example, a character like "<" no longer means "LESS-THAN SIGN"; it means the start of a tag or element that affects the document in some way.

On the formal end of the formatted text spectrum is markup, a family of languages for annotating a document in such a way that the annotations are syntactically distinguishable from the text. Markup languages are (reasonably) well-specified and tend to follow (mostly)

standardized syntax rules. Examples of markup languages include SGML, HTML, XML, and LaTeX. Standardized rules lead to interoperability between markup processors, but a skill requirement for new (human) users of the language that they learn these rules in order to do useful work. This imposition makes markup less accessible for non-technical users (i.e., users who are unwilling or unable to invest in the requisite skill development).

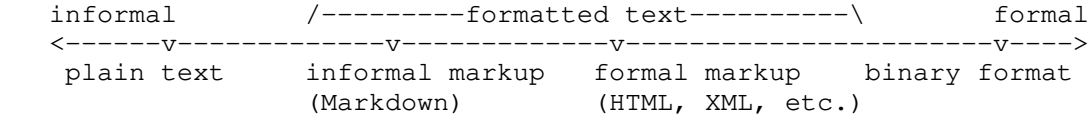


Figure 1: Degrees of Formality in Data Storage Formats for Text

On the informal end of the spectrum are lightweight markup languages. In comparison with formal markup like XML, lightweight markup uses simple syntax, and is designed to be easy for humans to enter with basic text editors. Markdown, the subject of this document, is an /informal/ plain text formatting syntax that is intentionally targeted at non-technical users (i.e., users upon whom little to no skill development is imposed) using unspecialized tools (i.e., text boxes). Jeff Atwood once described these informal markup languages as "humane" [HUMANE].

1.2. Markdown Design Philosophy

Markdown specifically is a family of syntaxes that are based on the original work of John Gruber with substantial contributions from Aaron Swartz, released in 2004 [MARKDOWN]. Since its release a number of web or web-facing applications have incorporated Markdown into their text entry systems, frequently with custom extensions. Fed up with the complexity and security pitfalls of formal markup languages (e.g., HTML5) and proprietary binary formats (e.g., commercial word processing software), yet unwilling to be confined to the restrictions of plain text, many users have turned to Markdown for document processing. Whole toolchains now exist to support Markdown for online and offline projects.

Informality is a bedrock premise of Gruber's design. Gruber created Markdown after disastrous experiences with strict XML and XHTML processing of syndicated feeds. In Mark Pilgrim's "thought experiment", several websites went down because one site included invalid XHTML in a blog post, which was automatically copied via trackbacks across other sites [DIN2MD]. These scenarios led Gruber to believe that clients (e.g., web browsers) SHOULD try to make sense of data that they receive, rather than rejecting data simply because it fails to adhere to strict, unforgiving standards. (In [DIN2MD],

Gruber compared Postel's Law [RFC0793] with the XML standard, which says: "Once a fatal error is detected [...] the processor MUST NOT continue normal processing" [XML1.0-5].) As a result, there is no such thing as "invalid" Markdown; there is no standard demanding adherence to the Markdown syntax; there is no governing body that guides or impedes its development. If the Markdown syntax does not result in the "right" output (defined as output that the author wants, not output that adheres to some dictated system of rules), Gruber's view is that the author either should keep on experimenting, or should change the processor to address the author's particular needs (see [MARKDOWN] Readme and [MD102b8] perldoc; see also [CATPICS]).

1.3. Uses of Markdown

Since its introduction in 2004, Markdown has enjoyed remarkable success. Markdown works for users for three key reasons. First, the markup instructions (in text) look similar to the markup that they represent; therefore the cognitive burden to learn the syntax is low. Second, the primary arbiter of the syntax's success is *running code*. The tool that converts the Markdown to a presentable format, and not a series of formal pronouncements by a standards body, is the basis for whether syntactic elements matter. Third, Markdown has become something of an Internet meme [INETMEME], in that Markdown gets received, reinterpreted, and reworked as additional communities encounter it. There are communities that are using Markdown for scholarly writing [OCCASION], for screenplays [FOUNTAIN], and even for mathematical formulae [MATHDOWN]. Clearly, a screenwriter has no use for specialized Markdown syntax for mathematicians; likewise, mathematicians do not need to identify characters or props in common ways. The overall gist is that all of these communities can take the common elements of Markdown (which are rooted in the common elements of HTML circa 2004) and build on them in ways that best fit their needs.

1.4. Uses of Labeling Markdown Content as text/markdown

The primary purpose of an Internet media type is to label "content" on the Internet, as distinct from "files". Content is any computer-readable format that can be represented as a primary sequence of octets, along with type-specific metadata (parameters) and type-agnostic metadata (protocol dependent). From this description, it is apparent that appending ".markdown" to the end of a filename is not a sufficient means to identify Markdown. Filenames are properties of files in file systems, but Markdown frequently exists in databases or content management systems (CMSes) where the file metaphor does not apply. One CMS [RAILFROG] uses media types to select appropriate processing, so a media type is necessary for the safe and

interoperable use of Markdown.

Unlike complete HTML documents, [MDSYNTAX] provides no means to include metadata into the content stream. Several derivative flavors have invented metadata incorporation schemes (e.g., [MULTIMD]), but these schemes only address specific use cases. In general, the metadata must be supplied via supplementary means in an encapsulating protocol, format, or convention. The relationship between the content and the metadata is not directly addressed here or in [MDMTREG]; however, by identifying Markdown with a media type, Markdown content can participate as a first-class citizen with a wide spectrum of metadata schemes.

Finally, registering a media type through the IETF process is not trivial. Markdown can no longer be considered a "vendor"-specific innovation, but the registration requirements even in the vendor tree have proven to be overly burdensome for most Markdown implementers. Moreover, registering hundreds of Markdown variants with distinct media types would impede interoperability: virtually all Markdown content can be processed by virtually any Markdown processor, with varying degrees of success. The goal of [MDMTREG] is to reduce all of these burdens by having one media type that accommodates diversity and eases registration.

1.5. Definitions

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].

Since Markdown signifies a family of related formats with varying degrees of formal documentation and implementation, this specification uses the term "variant" to identify such formats.

2. Strategies for Preserving Media Type and Parameters

The purpose of this document and [MDMTREG] is to promote interoperability between different Markdown-related systems, preserving the author’s intent. While [MARKDOWN] was designed by Gruber in 2004 as a simple way to write blog posts and comments, as of 2014 Markdown and its derivatives are rapidly becoming the formats of record for many communities and use cases. While an individual member of (or software tool for) a community can probably look at some "Markdown" and declare its meaning intuitively obvious, software systems in different communities (or different times) need help. [MDSYNTAX] does not have a signaling mechanism like <!DOCTYPE>, so tagging Markdown internally is simply out of the question. Once tags or metadata are introduced, the content is no longer "just" Markdown.

Some commentators have suggested that an in-band signaling mechanism, such as in Markdown link definitions at the top of the content, could be used to signal the variant. Unfortunately this signaling mechanism is incompatible with other Markdown variants (e.g., [PANDOC]) that expect their own kinds of metadata at the top of the file. Markdown content is just a stream of text; the semantics of that text can only be furnished by context.

The media type and variant parameter in [MDMTREG] furnish this missing context, while allowing for additional extensibility. This section covers strategies for how an application might preserve metadata when it leaves the domain of IETF protocols.

[MDMTREG] only defines two parameters: the charset parameter (required for all text/* media types) and the variant parameter. [RFC6657] provides guidance on character set parameter handling. The variant parameter provides a simple identifier--nothing less or more. Variants are allowed to define additional parameters when sent with the text/markdown media type; the variant can also introduce control information into the textual content stream (such as via a metadata block). Neither [MDMTREG] nor this specification recommend any particular approach. However, the philosophy behind [MDMTREG] is to preserve formats rather than create new ones, since supporting existing toolchains is more realistic than creating novel ones that lack traction in the Markdown community.

2.1. Map to Filename and Attributes

This strategy is to map the media type, variant, and parameters to "attributes" or "forks" in the local convention. Firstly, Markdown content saved to a file should have an appropriate file extension ending in .md or .markdown, which serves to disambiguate it from other kinds of files. The character repertoire of variant identifiers in [MDMTREG] is designed to be compatible with most filename conventions. Therefore, a recommended strategy is to record the variant identifier as the prefix to the file extension. For example, for [PANDOC] content, a file could be named "example.pandoc.markdown".

Many filesystems are case-sensitive or case-preserving; however, file extensions tend to be all-lowercase. This document takes no position on whether variant identifiers should be case-preserved or all-lowercase when Markdown content is written to a file. However, when the variant identifier is read to influence operational behavior, it needs to be compared case-insensitively.

Many modern filesystems support "extended attributes", "alternate data streams", or "resource forks". Some version control systems

support named properties. If the variant defines additional parameters, these parameters should be stored in these resources, where the parameter name includes the name of the resource, and the parameter value is the value of the resource (data in the resource), preferably UTF-8 encoded (unless the parameter definition explicitly defines a different encoding or repertoire). The variant identifier itself should be stored in a resource with a name including the term "variant" (possibly including other decorations to avoid namespace collisions).

2.2. Store Headers in Adjacent File

This strategy is to save the Markdown content in a first file, and to save the metadata (specifically the Content-Type: header) in a second file with a filename that is rationally related to the first filename. For example, if the first file is named "readme.markdown", the second file could be named "readme.markdown.headers". (If stored in a database, the analogy would be to store the metadata in a second table with a field that is a key to the first table.) This header file has the media type "message/global-headers" [RFC6533] (".u8hdr" suggestion notwithstanding).

2.3. "Arm" Content with MIME Headers

This strategy is to save the Markdown content along with its headers in a file, "arming" the content by prepending the MIME headers (specifically the Content-Type: header). It should be appreciated that the file is no longer a "Markdown file"; rather, it is an Internet Message Format file (e.g., [RFC5322]) with a Markdown content part. Therefore, the file should have an Internet message extension (e.g., ".eml", ".msg", or ".u8msg"), not a Markdown extension (e.g., ".md" or ".markdown").

2.4. Create a Local Batch Script

This strategy is to translate the processing instructions inferred from the Content-Type and other parameters (e.g., Content-Disposition) into a sequence of commands in the local convention, storing those commands in a batch script. For example, when a MIME-aware client stores some Markdown to disk, the client can save a Makefile in the same directory with commands that are appropriate (and safe) for the local system.

2.5. Process the Markdown in Advance

This strategy is to process the Markdown into the formal markup, before a recipient receives it, which eliminates ambiguities. Once the Markdown is processed into (for example) valid XHTML, an application can save a file as "doc.xhtml" or can send MIME content as application/xhtml+xml with no further loss of metadata. While unambiguous, this process may not be reversible.

2.6. Rely on Context

This last strategy is to use or create context to determine how to interpret the Markdown. For example, Markdown content that is of the Fountain.io type [FOUNTAIN] could be saved with the filename "script.fountain" instead of "script.markdown". Alternatively, scripts could be stored in a "/screenplays" directory while other kinds of Markdown could be stored elsewhere. For reasons that should be intuitively obvious, this method is the most error-prone. "Context" can be easily lost over time, and the trend of passing Markdown between systems--taking them *out* of context--is increasing.

2.7. Specific Strategies

2.7.1. Subversion

This subsection covers a preservation strategy in Subversion [SVN], a common client-server version control system.

Subversion supports named properties. The "svn:mime-type" property duplicates the entire Content-Type header, so parameters SHOULD be stored there (Section 2.1). The filename SHOULD be consistent with this Content-Type header, i.e., the extension SHOULD be the variant identifier plus ".markdown" (Section 2.1).

2.7.2. Git

This subsection covers a preservation strategy in Git [GIT], a common distributed version control system.

Versions of Git as of the time of this writing do not support arbitrary metadata storage; however, third-party projects add this support.

If Git is used without a metadata storage service, then a reasonable strategy is to include the variant identifier in the filename (Section 2.1). The default text encoding SHOULD be UTF-8. For other or different properties, a header file SHOULD be recorded alongside

the Markdown file (Section 2.2).

If a metadata storage service is used with Git, then use a convention that is most analogous to the service. For example, the "metastore" project emulates extended attributes (xattrs) of a POSIX-like system, so whatever "xattr" methodology is developed would be usable with metastore and Git.

3. Registration Templates for Common Markdown Syntaxes

The purpose of this section is to register certain syntaxes in the Markdown Syntaxes Registry [MDMTREG] because they illustrate particularly interesting use cases or are broadly applicable to the Internet community; thus, these syntaxes would benefit from the level of review associated with publication as IETF documents.

3.1. MultiMarkdown

Identifier: MultiMarkdown

Name: MultiMarkdown

Description:
MultiMarkdown (MMD) is a superset of "Original". It adds multiple syntax features (tables, footnotes, and citations, to name a few), and is intended to output to various formats. Additionally, it builds in "smart" typography for various languages (proper left- and right-sided quotes, for example).

Additional Parameters:
 options: String with zero or more of the following WSP-delimited tokens:

```
"memoir" / "beamer"
"full" / "snippet"
"process-html"
"random-footnote-identifiers"
"accept"
"reject"
"nosmart"
"nonotes"
"nolabels"
"nomask"
```

The meanings of these tokens are defined in the MultiMarkdown documentation.

References:

<http://fletcher.github.io/MultiMarkdown-4/syntax>

Contact Information:
(individual) Fletcher T. Penney <fletcher@fletcherpenney.net>
<http://fletcherpenney.net/multimarkdown/>

3.2. GitHub Flavored Markdown

Identifier: GFM

Name: GitHub Flavored Markdown

Description:
"Original" with the following differences:
1. Multiple underscores in words
2. URL (URI) autolinking
3. Strikethrough
4. Fenced code blocks
5. Syntax highlighting
6. Tables (- for rows; | for columns; : for alignment)
7. Only some HTML allowed; sanitization is integral to the format

References:
<https://help.github.com/articles/github-flavored-markdown/>
<https://github.com/github/markup/tree/master#html-sanitization>

Contact Information:
(corporate) GitHub, Inc. <https://github.com/contact>

3.3. Pandoc

Identifier: pandoc

Name: Pandoc

Description:
Markdown is designed to be easy to write and to read: the content should be publishable as-is, as plain text, without looking like it has been marked up with tags or formatting instructions. Yet whereas "Original" has HTML generation in mind, pandoc is designed for multiple output formats. Thus, while pandoc allows the embedding of raw HTML, it discourages it, and provides other, non-HTMLish ways of representing important document elements like definition lists, tables, mathematics, and footnotes.

Additional Parameters:
extensions: String with an optional starting syntax token, followed

by a "+" and "-" delimited list of extension tokens. "+" preceding an extension token turns the extension on; "-" turns the extension off. The starting syntax tokens are "markdown", "markdown_strict", "markdown_phpextra", and "markdown_github". If no starting syntax token is given, "markdown" is assumed. The extension tokens include:

Extensions to turn off (on by default):

escaped_line_breaks
blank_before_header
header_attributes
auto_identifiers
implicit_header_references
blank_before_blockquote
fenced_code_blocks
fenced_code_attributes
line_blocks
fancy_lists
startnum
definition_lists
example_lists
table_captions
simple_tables
multiline_tables
grid_tables
pipe_tables
pandoc_title_block
yaml_metadata_block
all_symbols_escapable
intraword_underscores
strikeout
superscript
subscript
inline_code_attributes
tex_math_dollars
raw_html
markdown_in_html_blocks
native_divs
native_spans
raw_tex
latex_macros
implicit_figures
footnotes
inline_notes
citations

Extensions to turn on (off by default):

```
                    lists_without_preceding_blankline
                    hard_line_breaks
                    ignore_line_breaks
                    tex_math_single_backslash
                    tex_match_double_backslash
                    markdown_attribute
                    mmd_title_block
                    abbreviations
                    autolink_bare_uris
                    ascii_identifiers
                    link_attributes
                    mmd_header_identifiers
                    compact_definition_lists
```

Fragment Identifiers:
Pandoc defines fragment identifiers using the <id> in the {#<id> .class ...} production (PHP Markdown Extra attribute block). This syntax works for Header Identifiers and Code Block Identifiers.

References:
<http://johnmacfarlane.net/pandoc/README.html#pandocs-markdown>

Contact Information:
(individual) Prof. John MacFarlane <jgm@berkeley.edu>
<http://johnmacfarlane.net/>

3.4. Fountain (Fountain.io)

Identifier: Fountain

Name: Fountain

Description:
Fountain is a simple markup syntax for writing, editing and sharing screenplays in plain, human-readable text. Fountain allows you to work on your screenplay anywhere, on any computer or tablet, using any software that edits text files.

Fragment Identifiers:
See <http://fountain.io/syntax#section-titlepage> and <http://fountain.io/syntax#section-sections>. In the following fragment identifiers, the <key> and <sec*> productions MUST have "/" characters percent-encoded.

```
#/       Title Page (acts as metadata).
#/<key>  Title Page; <key> is the key string.
#<sec1> *("/" <secn>)
         Section or subsection. The <sec1>..<secn>
```

productions are the text of the Section line,
with whitespace trimmed from both ends.
Sub-sections (sections with multiple # at
at the beginning of the line in the source)
are addressed hierarchically by preceding
the sub-section with higher-order
sections. If the section hierarchy "skips",
e.g., # to ###, use a blank section name,
e.g., #Section/ACT%20I//PATIO%20SCENE.

References:
<http://fountain.io/syntax>

Contact Information:
(individual) Stu Maschwitz <http://prolost.com/>
(individual) John August <http://johnaugust.com/>

3.5. CommonMark

Identifier: CommonMark

Name: CommonMark

Description:
CommonMark is a standard, unambiguous syntax specification for Markdown, along with a suite of comprehensive tests to validate Markdown implementations against this specification. The maintainers believe that CommonMark is necessary, even essential, for the future of Markdown.

Compared to "Original", CommonMark is much longer and in a few instances contradicts "Original" based on seasoned experience. Although CommonMark specifically does not mandate any particular encoding for the input content, CommonMark draws in more of Unicode, UTF-8, and HTML (including HTML5) than "Original".

This registration always refers to the latest version or an unspecified version (receiver's choice). Version 0.13 of the CommonMark specification was released 2014-12-10.

References:
<http://spec.commonmark.org/>

Contact Information:
(individual) John MacFarlane <jgm@berkeley.edu>
(individual) David Greenspan <david@meteor.com>
(individual) Vicent Marti <vicent@github.com>
(individual) Neil Williams <neil@reddit.com>

(individual) Benjamin Dumke-von der Ehe <ben@stackexchange.com>
(individual) Jeff Atwood <jatwood@codinghorror.com>

3.6. kramdown-rfc2629 (Markdown for RFCs)

Identifier: kramdown-rfc2629

Name: Markdown for RFCs

Description:
kramdown is a markdown parser by Thomas Leitner, which has a number of backends for generating HTML, Latex, and Markdown again. kramdown-rfc2629 is an additional backend to that: It allows the generation of XML2RFC XML markup (also known as RFC 2629 compliant markup).

References:
<https://github.com/cabo/kramdown-rfc2629>

Contact Information:
(individual) Carsten Bormann <cabo@tzi.org>

3.7. rfc7328 (Pandoc2rfc)

Identifier: rfc7328

Name: Pandoc2rfc

Description:
Pandoc2rfc allows authors to write in "pandoc" that is then transformed to XML and given to xml2rfc. The conversions are, in a way, amusing, as we start off with (almost) plain text, use elaborate XML, and end up with plain text again.

References:
RFC 7328
<https://github.com/miekg/pandoc2rfc>

Contact Information:
(individual) R. (Miek) Gieben <miek@google.com>

3.8. PHP Markdown Extra

Identifier: Extra

Name: Markdown Extra

Description:
Markdown Extra is an extension to PHP Markdown implementing some

features currently not available with the plain Markdown syntax. Markdown Extra is available as a separate parser class in PHP Markdown Lib. Other implementations include Maruku (Ruby) and Python Markdown. Markdown Extra is supported in several content management systems, including Drupal, TYPO3, and MediaWiki.

Fragment Identifiers:
Markdown Extra defines fragment identifiers using the <id> in the {#<id> .class ...} production (attribute block). This syntax works for headers, fenced code blocks, links, and images.

References:
<https://michelf.ca/projects/php-markdown/extra/>

Contact Information:
(individual) Michel Fortin <michel.fortin@michelf.ca>

4. Examples for Common Markdown Syntaxes

This section provides examples of the variants in Appendix C.

4.1. MultiMarkdown

```
Title:    Example of MultiMarkdown
Keywords: IETF, example, footnotes


# MutliMarkdown Example #

MultiMarkdown supports several cool features, as well as
several output formats:
*   HTML
*   PDF
*   OpenDocument
*   OPML
*   LaTeX

## Footnotes ##

Footnotes are described in the
MultiMarkdown Syntax Guide.[^somesamplefootnote]

[^somesamplefootnote]: Here is the text of the footnote itself.
```

Figure 1: MultiMarkdown Example

4.2. GitHub Flavored Markdown

```
# Start Out #

GFM is like regular Markdown with a few extra features. For example,
http://www.example.com/ will get auto-linked. ~~This is strikethrough
text, demarked by the double tildes.~~

```
function test() {
  return "notice this feature?");
}
```

Table Alignments

Left	Center	Right
cats	Paxton	$1600
dogs	Ruff	$30
zebras	Stripes	$20900
```

Figure 2: GitHub Flavored Markdown Example

4.3. Pandoc

```
% Pandoc User's Guide
% John MacFarlane
% August 30, 2014

Synopsis {#syn}
========

pandoc [*options*] [*input-file*]...

Description {#desc}
===========

Pandoc is a [Haskell] library for converting from one markup format to
another, and a command-line tool that uses this library.

#### Extension: `header_attributes` #### {#ext-header-attributes}

Headers can be assigned attributes using this syntax at the end
of the line containing the header text:

    {#identifier .class .class key=value key=value}

Thus, for example, the following headers will all be assigned
the identifier `foo`:
```

```
    # My header {#foo}

    ## My header ##    {#foo}

    My other header   {#foo}
    ---------------
```

Figure 3: Pandoc Example

4.4. Fountain (Fountain.io)

```
INT. BOXCAR - MOVING - DAY
?AGENT MORTIMER lies bleeding in the corner. The car ROCKS gently.
Mortimer pulls out his cell phone and dials.

MORTIMER?
Come on. Pick up.

CUT TO:?
ext. hotel bar - day?
A fiercely gorgeous brunette sips the last of something from a
rocks glass. This is REBECCA.

Behind her, a dark FIGURE approaches. She seems not to notice.

REBECCA?(to Bartender)
Ritenhouse, neat.

FIGURE (O.S.) ^
Ritenhouse, neat.

She turns to find the source of the voice.

FIGURE
Excellent choice.

Before she can reply, her phone RINGS.?

> INTERCUT WITH:?

.THE BOXCAR

Where MORTIMER is just barely holding on to life.
```

Figure 4: Fountain Example

4.5. CommonMark

```
CommonMark is like Markdown.

Here are some entity names that you can use with CommonMark: ` 
&amp; &copy; &AElig; &Dcaron; &frac34; &HilbertSpace; &DifferentialD;
&ClockwiseContourIntegral;`

You can see more at [the CommonMark website](http://commonmark.org/
"CommonMark").

- foo
***
- bar

Tildes can be used for fenced code blocks:

~~~
<
 >
~~~
```

Figure 5: CommonMark Example

4.6. kramdown-rfc2629 (Markdown for RFCs)

```
---
title: STUN/TURN using PHP in Despair
abbrev: STuPiD-excerpt
docname: draft-hartke-xmpp-stupid-excerpt-00
date: 2009-07-05
category: info

ipr: trust200902
area: General
workgroup: XMPP Working Group
keyword: Internet-Draft

stand_alone: yes
pi: [toc, sortrefs, symrefs]

author:
 -
    ins: K. Hartke
    name: Klaus Hartke
    email: example@tzi.org

normative:
  RFC2119:
```

```
informative:
  RFC5389:
  STUNT:
    target: http://www.example.com/oob
    title: STUNT & out-of-band channels
    author:
      name: Robbie Hanson
      ins: R. Hanson
    date: 2007-09-17



--- abstract

NAT (Network Address Translator) Traversal may require TURN
(Traversal Using Relays around NAT) functionality in certain
cases that are not unlikely to occur.  There is little
incentive to deploy TURN servers, except by those who need
them&#x2014;who may not be in a position to deploy a new protocol
on an Internet-connected node, in particular not one with
deployment requirements as high as those of TURN.

--- middle

Introduction        {#problems}
============

"STUN/TURN using PHP in Despair" is a highly deployable
protocol for obtaining TURN-like functionality, while also
providing the most important function of STUN
{{RFC5389}}.


The Need for Standardization   {#need}
----------------------------

Having one standard form of STuPiD service instead of one
specific to each kind of client also creates an incentive
for optimized implementations.
```

```
~~~~~~~~~~


        STuPiD   ````````````````````````````,
        Script   <----------------------------. ,
                                              | ,
          ^ ,                                 | ,
          | ,                                 | ,
    (1)   | ,                                 | ,  (3)
    POST  | ,                                 | ,  GET
          | ,                                 | ,
          | v                                 | v

        Peer A   ----------------------->   Peer B
                          (2)
                    out-of-band
                    Notification
~~~~~~~~~~
```

{: #figops title="STuPiD Protocol Operation"}

Terminology {#Terminology}

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 {{RFC2119}} and indicate requirement levels for compliant STuPiD implementations.

--- back

Sample Implementation {#impl}

```
~~~~~~~~~~
<?php
header("Cache-Control: no-cache, must-revalidate");
header("Expires: Sat, 26 Jul 1997 05:00:00 GMT");
header("Content-Type: application/octet-stream");

?>
~~~~~~~~~~
```

{: #figimpl title="STuPiD Sample Implementation"}

Figure 6: Markdown for RFCs Example

4.7. rfc7328 (Pandoc2rfc)

Pandoc2rfc expects multiple files as input. The following figure is example of "middle.mkd".

```
# Introduction

<?rfc toc="yes"?>
<?rfc symrefs="yes"?>
<?rfc sortrefs="yes"?>
<?rfc subcompact="no"?>
<?rfc compact="yes"?>
<?rfc comments="yes"?>

This document presents a technique for using Pandoc syntax as a source
format for documents in the Internet-Drafts (I-Ds) and Request
for Comments (RFC) series.

This version is adapted to work with ‘xml2rfc‘ version 2.x.

Pandoc is an "almost plain text" format and therefore particularly
well suited for editing RFC-like documents.

> Note: this document is typeset in Pandoc.

> NB: this is mostly text to test Pandoc2rfc, the canonical
> documentation is [draft-gieben-pandoc2rfc][p2r].

[p2r]: http://tools.ietf.org/html/draft-gieben-pandoc2rfc-01

# Pandoc to RFC

> Pandoc2rfc -- designed to do the right thing, until it doesn’t.

When writing [](#RFC4641) we directly wrote the
XML. Needless to say it was tedious even thought the XML of
[xml2rfc](http://xml.resource.org/experimental) is very "light".
The [latest version of xml2rfc version 2 can be found
here](http://pypi.python.org/pypi/xml2rfc/).
```

Figure 7: Pandoc2rfc Example (middle.mkd).

5. IANA Considerations

IANA is asked to register the syntaxes specified in Section 3 in the Markdown Variants Registry.

6. Security Considerations

See the respective syntax descriptions and output media type registrations for their respective security considerations.

7. References

7.1. Normative References

[MARKDOWN] Gruber, J., "Daring Fireball: Markdown", December 2004, <http://daringfireball.net/projects/markdown/>.

[MDSYNTAX] Gruber, J., "Daring Fireball: Markdown Syntax Documentation", December 2004, <http://daringfireball.net/projects/markdown/syntax>.

[MDMTREG] Leonard, S., "The text/markdown Media Type", draft-ietf-appsawg-text-markdown-11 (work in progress), September 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.

[RFC6657] Melnikov, A. and J. Reschke, "Update to MIME regarding "charset" Parameter Handling in Textual Media Types", RFC 6657, July 2012.

7.2. Informative References

[UNICODE] The Unicode Consortium, "The Unicode Standard, Version 8.0.0", The Unicode Consortium, August 2015.

[HUMANE] Atwood, J., "Is HTML a Humane Markup Language?", May 2008, <http://blog.codinghorror.com/is-html-a-humane-markup-language/>.

[DIN2MD] Gruber, J., "Dive Into Markdown", March 2004, <http://daringfireball.net/2004/03/dive_into_markdown>.

[MD102b8] Gruber, J., "[ANN] Markdown.pl 1.0.2b8", May 2007, <http://six.pairlist.net/pipermail/markdown-discuss/2007-May/000615.html>, <http://daringfireball.net/projects/downloads/Markdown_1.0.2b8.tbz>.

[CATPICS] Gruber, J. and M. Arment, "The Talk Show: Ep. 88: 'Cat

Pictures' (Side 1)", July 2014, <http://daringfireball.net/thetalkshow/2014/07/19/ep-088>.

[INETMEME] Solon, O., "Richard Dawkins on the internet's hijacking of the word 'meme'", June 2013, <http://www.wired.co.uk/news/archive/2013-06/20/richard-dawkins-memes>, <http://www.webcitation.org/6HzDGE9Go>.

[MULTIMD] Penney, F., "MultiMarkdown", April 2014, <http://fletcherpenney.net/multimarkdown/>.

[PANDOC] MacFarlane, J., "Pandoc", 2014, <http://johnmacfarlane.net/pandoc/>.

[RAILFROG] Railfrog Team, "Railfrog", April 2009, <http://railfrog.com/>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC6533] Hansen, T., Ed., Newman, C. and A. Melnikov, "Internationalized Delivery Status and Disposition Notifications", RFC 6533, February 2012.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[XML1.0-5] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <http://www.w3.org/TR/2008/REC-xml-20081126#dt-fatal>.

[OCCASION] Shieber, S., "Switching to Markdown for scholarly article production", August 2014, <http://blogs.law.harvard.edu/pamphlet/2014/08/29/switching-to-markdown-for-scholarly-article-production/>.

[FOUNTAIN] Maschwitz, S. and J. August, "Fountain | A markup language for screenwriting.", 2014, <http://fountain.io/>.

[MATHDOWN] Cherniavsky-Paskin, B., "math in markdown", 2015, <https://github.com/cben/mathdown/wiki/math-in-markdown>.

[SVN] Apache Subversion, August 2015, <https://subversion.apache.org/>.

[GIT] Git, July 2015, <http://git-scm.com/>.

Author's Address

Sean Leonard
Penango, Inc.
5900 Wilshire Boulevard
21st Floor
Los Angeles, CA 90036
USA

EMail: dev+ietf@seantek.com
URI: http://www.penango.com/