The 'acct' URI Scheme

draft-ietf-appsawg-acct-uri-07

Abstract

This document defines the 'acct' Uniform Resource Identifier (URI) scheme as a way to identify a user’s account at a service provider, irrespective of the particular protocols that can be used to interact with the account.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 27, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
Internet-Draft            The 'acct' URI Scheme             January 2014

Table of Contents

1. Introduction .......................... 2
2. Terminology ............................ 2
3. Rationale .............................. 2
4. Definition .............................. 3
5. Security Considerations .................. 4
6. Internationalization Considerations ........ 5
7. IANA Considerations ..................... 5
8. References ............................. 7
Appendix A. Acknowledgements .................. 8
Author's Address ........................... 8

1. Introduction

Existing Uniform Resource Identifier (URI) schemes that enable interaction with, or that identify resources associated with, a user's account at a service provider are tied to particular services or application protocols. Two examples are the 'mailto' scheme (which enables interaction with a user's email account) and the 'http' scheme (which enables retrieval of web files controlled by a user or interaction with interfaces providing information about a user). However, there exists no URI scheme that generically identifies a user's account at a service provider without specifying a particular protocol to use when interacting with the account. This specification fills that gap.

2. Terminology

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

3. Rationale

During formalization of the WebFinger protocol [RFC7033], much discussion occurred regarding the appropriate URI scheme to include when specifying a user's account as a web link [RFC5988]. Although both the 'mailto' [RFC5988] and 'http' [RFC2616] schemes were proposed, not all service providers offer email services or web interfaces on behalf of user accounts (e.g., a microblogging or instant messaging provider might not offer email services, or an enterprise might not offer HTTP interfaces to information about its employees). Therefore, the participants in the discussion recognized that it would be helpful to define a URI scheme that could be used to generically identify a user's account at a service provider, irrespective of the particular application protocols used to interact
with the account. The result was the ‘acct’ URI scheme defined in this document.

(Note that a user is not necessarily a human; it could be an automated application such as a bot, a role-based alias, etc. However, an ‘acct’ URI is always used to identify something that has an account at a service, not the service itself.)

4. Definition

The syntax of the ‘acct’ URI scheme is defined under Section 7 of this document. Although ‘acct’ URIs take the form "user@host", the scheme is designed for the purpose of identification instead of interaction (regarding this distinction, see Section 1.2.2 of [RFC3986]). The "Internet resource" identified by an ‘acct’ URI is a user's account hosted at a service provider, where the service provider is typically associated with a DNS domain name. Thus a particular ‘acct’ URI is formed by setting the "user" portion to the user's account name at the service provider and by setting the "host" portion to the DNS domain name of the service provider.

Consider the case of a user with an account name of "foobar" on a microblogging service "status.example.net". It is taken as convention that the string "foobar@status.example.net" designates that account. This is expressed as a URI using the ‘acct’ scheme as "acct:foobar@status.example.net".

A common scenario is for a user to register with a service provider using an identifier (such as an email address) that is associated with some other service provider. For example, a user with the email address "juliet@capulet.example" might register with a commerce website whose domain name is "shoppingsite.example". In order to use her email address as the localpart of the ‘acct’ URI, the at-sign character (U+0040) needs to be percent-encoded as described in [RFC3986]. Thus the resulting ‘acct’ URI would be "acct:juliet%40capulet.example@shoppingsite.example".

It is not assumed that an entity will necessarily be able to interact with a user’s account using any particular application protocol, such as email; to enable such interaction, an entity would need to use the appropriate URI scheme for such a protocol, such as the ‘mailto’ scheme. While it might be true that the ‘acct’ URI minus the scheme name (e.g., "user@example.com" derived from "acct:user@example.com") can be reached via email or some other application protocol, that fact would be purely contingent and dependent upon the deployment practices of the provider.)
Because an ‘acct’ URI enables abstract identification only and not interaction, this specification provides no method for dereferencing an ‘acct’ URI on its own, e.g., as the value of the ‘href’ attribute of an HTML anchor element. For example, there is no behavior specified in this document for an ‘acct’ URI used as follows:

<a href='acct:bob@example.com'>find out more</a>

Any protocol that uses ‘acct’ URIs is responsible for specifying how an ‘acct’ URI is employed in the context of that protocol (in particular, how it is dereferenced or resolved; see [RFC3986]). As a concrete example, an "Account Information" application of the WebFinger protocol [RFC7033] might take an ‘acct’ URI, resolve the host portion to find a WebFinger server, and then pass the ‘acct’ URI as a parameter in a WebFinger HTTP request for metadata (i.e., web links [RFC5988]) about the resource. For example:

GET /.well-known/webfinger?resource=acct%3Abob%40example.com HTTP/1.1

The service retrieves the metadata associated with the account identified by that URI and then provides that metadata to the requesting entity in an HTTP response.

If an application needs to compare two ‘acct’ URIs (e.g., for purposes of authentication and authorization), it MUST do so using case normalization and percent-encoding normalization as specified in Sections 6.2.2.1 and 6.2.2.2 of [RFC3986].

5. Security Considerations

Because the ‘acct’ URI scheme does not directly enable interaction with a user’s account at a service provider, direct security concerns are minimized.

However, an ‘acct’ URI does provide proof of existence of the account; this implies that harvesting published ‘acct’ URIs could prove useful to spammers and similar attackers, for example if they can use an ‘acct’ URI to leverage more information about the account (e.g., via WebFinger) or if they can interact with protocol-specific URIs (such as ‘mailto’ URIs) whose user@host portion is the same as that of the ‘acct’ URI.

In addition, protocols that make use of ‘acct’ URIs are responsible for defining security considerations related to such usage, e.g., the risks involved in dereferencing an ‘acct’ URI, the authentication and authorization methods that could be used to control access to
personal data associated with a user’s account at a service, and methods for ensuring the confidentiality of such information.

The use of percent-encoding allows a wider range of characters in account names, but introduces some additional risks. Implementers are advised to disallow percent-encoded characters or sequences that would (1) result in space, null, control, or other characters that are otherwise forbidden, (2) allow unauthorized access to private data, or (3) lead to other security vulnerabilities.

6. Internationalization Considerations

As specified in [RFC3986], the ‘acct’ URI scheme allows any character from the Unicode repertoire [UNICODE] encoded as UTF-8 [RFC3629] and then percent-encoded into valid ASCII [RFC20]. Before applying any percent-encoding, an application MUST ensure the following about the string that is used as input to the URI-construction process:

- The userpart consists only of Unicode code points that conform to the PRECIS IdentifierClass specified in [I-D.ietf-precis-framework].
- The host consists only of Unicode code points that conform to the rules specified in [RFC5892].
- Internationalized domain name (IDN) labels are encoded as A-labels [RFC5890].

7. IANA Considerations

In accordance with the guidelines and registration procedures for new URI schemes [RFC4395], this section provides the information needed to register the ‘acct’ URI scheme.

7.1. URI Scheme Name

acct

7.2. Status

permanent

7.3. URI Scheme Syntax

The ‘acct’ URI syntax is defined here in Augmented Backus-Naur Form (ABNF) [RFC5234], borrowing the ‘host’, ‘pct-encoded’, ‘sub-delims’, ‘unreserved’ rules from [RFC3986]:

acctURI = "acct" ":" userpart "@" host
userpart = unreserved / sub-delims
0*( unreserved / pct-encoded / sub-delims )

Note that additional rules regarding the strings that are used as input to construction of ‘acct’ URIs further limit the characters that can be percent-encoded; see the Encoding Considerations as well as Section 6 of RFC XXXX. [Note to RFC Editor: please replace XXXX with the number issued to this document.]

7.4. URI Scheme Semantics

The ‘acct’ URI scheme identifies accounts hosted at service providers. It is used only for identification, not interaction. A protocol that employs the ‘acct’ URI scheme is responsible for specifying how an ‘acct’ URI is dereferenced in the context of that protocol. There is no media type associated with the ‘acct’ URI scheme.

7.5. Encoding Considerations

See Section 6 of RFC XXXX. [Note to RFC Editor: please replace XXXX with the number issued to this document.]

7.6. Applications/Protocols That Use This URI Scheme Name

At the time of this writing, only the WebFinger protocol uses the ‘acct’ URI scheme. However, use is not restricted to the WebFinger protocol, and the scheme might be considered for use in other protocols.

7.7. Interoperability Considerations

There are no known interoperability concerns related to use of the ‘acct’ URI scheme.

7.8. Security Considerations

See Section 5 of RFC XXXX. [Note to RFC Editor: please replace XXXX with the number issued to this document.]

7.9. Contact

Peter Saint-Andre, psaintan@cisco.com

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

7.11. References

None.

8. References

8.1. Normative References


8.2. Informative References

Appendix A. Acknowledgements

The `acct` URI scheme was originally proposed during work on the WebFinger protocol; special thanks are due to Blaine Cook, Brad Fitzpatrick, and Eran Hammer-Lahav for their early work on the concept (which in turn was partially inspired by work on Extensible Resource Identifiers at OASIS). The scheme was first formally specified in [RFC7033]; the authors of that specification (Paul Jones, Gonzalo Salgueiro, and Joseph Smarr) are gratefully acknowledged. Thanks are also due to Stephane Bortzmeyer, Melvin Carvalho, Martin Duerst, Graham Klyne, Barry Leiba, Subramanian Moonesamy, Evan Prodromou, James Snell, and various participants in the IETF APPSAWG for their feedback. Meral Shirazipour completed a Gen-ART review. Dave Cridland completed an AppsDir review, and is gratefully acknowledged for providing proposed text that was incorporated into Section 3 and Section 5. IESG comments from Richard Barnes, Adrian Farrel, Stephen Farrell, Barry Leiba, Pete Resnick, and Sean Turner also led to improvements in the specification.

Author’s Address

Peter Saint-Andre

Email: ietf@stpeter.im
The file URI Scheme
draft-ietf-appsawg-file-scheme-00

Abstract

This document specifies the "file" Uniform Resource Identifier (URI) scheme, replacing the definition in RFC 1738.

It attempts to document current practices, while at the same time defining a common core which is intended to interoperate across the broad spectrum of existing implementations.

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>.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 16, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents
1. Introduction

A file URI identifies a file on a particular file system. It can be used in discussions about the file, and if other conditions are met it can be dereferenced to directly access the file.

The file URI scheme is not coupled with a specific protocol. As such, there is no well-defined set of methods that can be performed on file URIs, nor a media type associated with them.

This document defines a syntax that is compatible with most extant implementations, while attempting to push towards a stricter subset of "ideal" constructs. In many cases it simultaneously acknowledges and deprecates some less common or outdated constructs.
1.1. History

The file URI scheme was first defined in [RFC1630], which, being an informational RFC, does not specify an Internet standard. The definition was standardised in [RFC1738], and the scheme was registered with the Internet Assigned Numbers Authority (IANA); however that definition omitted certain language included by former that clarified aspects such as:

- the use of slashes to denote boundaries between directory levels of a hierarchical file system; and
- the requirement that client software convert the file URI into a file name in the local file name conventions.

The Internet draft [I-D.draft-hoffman-file-uri] was written in an effort to keep the file URI scheme on standards track when [RFC1738] was made obsolete, but that draft expired in 2005. It enumerated concerns arising from the various, often conflicting implementations of the scheme. It serves as the spiritual predecessor of this document.

Additionally the WHATWG defines a living URL standard [WHATWG-URL], which includes algorithms for interpreting file URIs (as URLs).

1.2. Similar Technologies

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. A UNC filesystem selector string has three parts: host, share, and path; see: Appendix A. This document describes a means of translating between UNC filesystem selector strings and file URIs.

1.3. 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].

2. Syntax

The file URI syntax is defined here in Augmented Backus-Naur Form (ABNF) [RFC5234], including the core ABNF syntax rule "ALPHA" defined by that specification, and importing the "userinfo", "host", "path-absolute", and "query" rules from [RFC3986] (as updated by [RFC6874].)
file-URI = f-scheme ":" f-hier-part [ "?" query ]
f-scheme = "file"
f-hier-part = "//{{" auth-path
/ local-path
auth-path = [ f-auth ] path-absolute
/ unc-path
/ windows-path
f-auth = [ userinfo "@" ] host
local-path = path-absolute
/ windows-path
unc-path = 2*3"/" authority path-absolute
windows-path = drive-letter path-absolute
drive-letter = ALPHA [ drive-marker ]
drive-marker = ":" / "|

Note well: the "drive-marker" rule intentionally includes a bar character "|" even though that character is not part of either the unreserved or reserved character sets in [RFC3986], and thus would normally have to be percent-encoded to be included in a URI. This specification explicitly supports the parsing of otherwise invalid URIs - those with an unencoded bar character forming part of a DOS or Windows drive letter identifier - to facilitate parsing extant historical URIs, but new URIs of this form MUST NOT be generated.

The query field contains non-hierarchical data that, along with data in the path components (path-absolute, unc-path, or windows-path) serves to identify a resource. This is not commonly used in practice, but could be used to refer to a specific version of a file in a versioning file system, for example.

Systems exhibit different levels of case-sensitivity. Unless the file system is known to be case-insensitive, implementations MUST maintain the case of file and directory names when translating file URIs to and from the local system’s representation of file paths, and any systems or devices that transport file URIs MUST NOT alter the case of file URIs they transport.

The syntax definition above is necessarily different from those given in [RFC1630] and [RFC1738] because it depends on the generic syntax from [RFC3986] that post-dates all previous specifications.
It is intended to support file URIs that take the following forms:

Local files:
- "file:///path/to/file"
  A "traditional" file URI for a local file, with an empty authority. This is the most common format in use today, despite being technically incompatible with the definition in [RFC1738].
- "file:///c:/path/to/file"
  The traditional representation of a local file in a DOS- or Windows-based environment.
- "file:/path/to/file"
  A "modern" minimal representation of a local file in a UNIX-like environment, with no authority field and an absolute path that begins with a slash "/".
- "file:c:/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.
- "file:///c/path/to/file"
- "file:/c/path/to/file"
- "file:c/path/to/file"
  Representations of a local file in a DOS- or Windows-based environment, using alternative representations of drive letters. These are supported for compatibility with historical implementations, but deprecated by this specification.
- "file:/c:/path/to/file"
  A representation of a local file in a DOS- or Windows-based environment, with no authority field and a slash preceding the drive letter. This representation is less common than those above, and is deprecated by this specification.
Non-local files:

- "file://host.example.com/path/to/file"
  The ideal representation of a non-local file, with an explicit authority.

- "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.

- "file://///host.example.com/path/to/file"
  As above, with an extra slash between the empty authority and the transformed UNC string, conformant with the definition from [RFC1738]; see: exceptions in Section 3.2. This representation is deprecated by this specification. It is notably used by the Firefox web browser.

Dubious encodings:

- "file://c:/path/to/file"
- "file://c/path/to/file"
  An encoding that includes a Windows drive letter as the authority field. This encoding exists in some extant implementations, and is supported by the grammar for historical reasons. New URIs of this form SHOULD NOT be generated.

- "file:///c|/path/to/file"
- "file://c|/path/to/file"
- "file:/c|/path/to/file"
- "file:c|/path/to/file"
  Various generally invalid URIs that include a disallowed bar character "|" in the drive letter identifier. These encodings are supported by the grammar for historical reasons. As noted above, new URIs of this form MUST NOT be generated.

It also intentionally excludes URIs of the form:

- "file://auth.example.com//host.example.com/path/to/file"
An encoding that includes both a non-local authority, and a UNC string. The traditional implication was that the shared object described by the UNC string may only be accessed from the machine "auth.example.com".

3. Methods on file URIs

In the strictest terms, the only operations that can be performed on a file URI are translating it to and from a file path; subsequent methods are performed on the resulting file path, and depend entirely on the file system’s APIs.

For example, consider the POSIX "open()", "read()", and "close()" methods [POSIX] for reading a file’s contents into memory.

Some APIs allow file system methods to be invoked directly on file URIs, while others provide mappings to other similar methods, such as GET and PUT from the Hypertext Transfer Protocol (HTTP) [RFC7231].

The local file system API can only be used if the file URI has a blank (or absent) authority and the path, when transformed to the local system’s conventions, is not a UNC string. Note that this differs from the definition in [RFC1738] in that previously an authority containing the text "localhost" was used to refer to the local file system, but in this specification it translates to a UNC string referring to the host "localhost".

This specification does not define a mechanism for accessing files stored on non-local file systems.

3.1. Translating Local File Path to file URI

Below is an algorithmic description of the process used to convert a file path to an Internationalized Resource Identifier (IRI) [RFC3987], which can then be translated to a URI as per Section 3.1 of [RFC3987]; see: Section 4.

1. Resolve the file path to its fully qualified absolute form.

2. Initialise the URI with the "file:" scheme identifier.

3. If including an empty authority field, append the "/" sigil to the URI.
4. Append the root directory:
   * On a DOS- or Windows-based system, assign the drive letter (e.g. "c:"), as the first path segment, and append it to the URI, followed by a slash character "/".
   + If an empty authority was included at step 3, a slash "/" is prepended to the drive letter (e.g. "/c:"), to distinguish it from the authority.
   * On an OpenVMS Files-11 system, append a slash "/" to the URI, and encode the device name as the first segment as per step 5, below, except that the dollars sign character "$" is not treated as a reserved character in this segment.
   * On a UNIX-like system, append a slash "/" to the URI, to denote the root directory.

5. For each directory in the path after the root:
   1. Transform the directory name to a path segment ([RFC3986], Section 3.3) as per Section 2 of [RFC3986].
   2. Append the transformed segment and a delimiting slash character "/" to the URI.

6. If the path includes a file name:
   1. Transform the file name to a path segment as above.
   2. Append the transformed segment to the URI.

7. If any non-hierarchical data is required to identify the file (for example a version number in a versioning file system):
   1. Append a question mark character "?" to the URI.
   2. Transform the non-hierarchical data to a query field ([RFC3986], Section 3.4) as per Section 2 of [RFC3986].
   3. Append the transformed query field to the URI.
Examples:

<table>
<thead>
<tr>
<th>File Path</th>
<th>URIs (minimal, traditional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX-like:</td>
<td></td>
</tr>
<tr>
<td>/path/to/file</td>
<td>file:/path/to/file</td>
</tr>
<tr>
<td></td>
<td>file:///path/to/file</td>
</tr>
<tr>
<td>/path/to/dir/</td>
<td>file:/path/to/dir/</td>
</tr>
<tr>
<td></td>
<td>file:///path/to/dir/</td>
</tr>
<tr>
<td>DOS- or Windows-based:</td>
<td></td>
</tr>
<tr>
<td>c:\path\to\file.txt</td>
<td>file:c:/path/to/file.txt</td>
</tr>
<tr>
<td></td>
<td>file:///c:/path/to/file.txt</td>
</tr>
<tr>
<td>c:\path\to\dir\</td>
<td>file:c:/path/to/dir/</td>
</tr>
<tr>
<td></td>
<td>file:///c:/path/to/dir/</td>
</tr>
<tr>
<td>VMS Files-11:</td>
<td></td>
</tr>
<tr>
<td>::DISK1:[PATH.TO]FILE.TXT;2</td>
<td>file:///DISK1/PATH/TO/FILE.TXT?2</td>
</tr>
<tr>
<td></td>
<td>file:///DISK1/PATH/TO/FILE.TXT?2</td>
</tr>
</tbody>
</table>

Differences from RFC 1738

In [RFC1738] a file URL always started with the token "file://", followed by an authority and a "/". That "/" was not considered part of the path. This implies that the correct encoding for the above example file path in a UNIX-like environment would have been:

\[
\text{token} \quad + \text{authority} + \text{slash} + \text{path} \\
= "file://" + "" + "/" + "/path/to/file.txt" \\
= "file:///path/to/file.txt"
\]

However that construct was never used in practice, and in fact would have collided with the eventual encoding of UNC strings in URIs.

Exceptions

DOS/Windows:
Some implementations leave the leading slash off before the drive letter when authority is blank, e.g. "file://c:/..."

DOS/Windows:
Some implementations replace ":" with "|", and others leave it off completely. e.g. "file:///c|/..." or "file:///c/..."
3.2. Translating UNC String to file URI

A UNC filespace selector string can be directly translated to an Internationalized Resource Identifier (IRI) [RFC3987], which can then be translated to a URI as per Section 3.1 of [RFC3987]; see: Section 4.

1. Initialise the URI with the "file:" scheme identifier.

2. Append the authority:
   1. Append the "/" authority sigil to the URI.
   2. Append the hostname field of the UNC string to the URI.

3. Append the sharename:
   1. Transform the sharename to a path segment ([RFC3986], Section 3.3) as per Section 2 of [RFC3986].
   2. Append a delimiting slash character "/" and the transformed segment to the URI.

4. For each objectname:
   1. Transform the objectname to a path segment ([RFC3986], Section 3.3) as per Section 2 of [RFC3986].
   2. Append a delimiting slash character "/" and the transformed segment to the URI.

Example:

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

Exceptions

Many implementations accept the full UNC string in the URI path (with all backslashes "\\" converted to slashes "/"). Additionally, because [RFC1738] said that the first "/" after "file://[authority]" wasn’t part of the path, some implementations (including Firefox) require an additional slash before the UNC string.
For example:

Traditional:
file:///hostname/share/object/names
\_____\__________________________ /
Scheme Transformed UNC string

Firefox:
file:///hostname/share/object/names
\_____\__________________________ /
Scheme Transformed UNC string Extra slash

3.3. Translating Non-local File Path to file URI

Translating a non-local file path other than a UNC string to a file URI follows the same basic algorithm as for local files, above, except that the authority MUST refer to the network-accessible node that hosts the file.

For example, in a clustered OpenVMS Files-11 system the authority would contain the node name. Where the original node reference includes a username and password in an access control string, they MAY be transcribed into the userinfo field of the authority ([RFC3986], Section 3.2.1), security considerations (Section 5) notwithstanding.

3.4. Incompatible File Paths

Some conventional file path formats are known to be incompatible with the file URI scheme.

3.4.1. Namespaces

The Microsoft Windows API defines Win32 Namespaces [Win32-Namespace] 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.

4. Encoding

The encoding of a file URI depends on the file system. If the file system uses a known non-Unicode character encoding, the path SHOULD
be converted to a sequence of characters from the Universal Character
Set [ISO10646] normalized according to Normalization Form C (NFC)
[UTR15], before being translated to a file URI, and conversely a file
URI SHOULD be converted back to the file system’s native encoding
when translating to a file path.

Note that many modern file systems encode directory and file names
as arbitrary sequences of octets. In those cases, the
representation as an encoded string often depends on the user’s
localization settings, or defaults to UTF-8 [STD63].

When the file system’s encoding is not known the file URI SHOULD be
transported as an Internationalized Resource Identifier (IRI)
[RFC3987].

Example: file IRI:

<table>
<thead>
<tr>
<th>Bytes of file IRI in a UTF-8 document:</th>
</tr>
</thead>
<tbody>
<tr>
<td>66 69 6c 65 3a 43 3a 2f 72 65 c3 a7 75 2e 74 78 74</td>
</tr>
<tr>
<td>f i l e : c : / r e ( c ) u . t x t</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interpretation:</th>
</tr>
</thead>
</table>
| A file named "recu.txt" with a cedilla on the "c", in the
directory "C:" of a DOS or Windows file system. |

| Character value sequences of file paths, for various file system
encodings: |
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>o UTF-16 (e.g. NTFS):</td>
</tr>
<tr>
<td>0043 003a 005c 0072 0065 00e7 0075 002e 0074 0078 0074</td>
</tr>
<tr>
<td>o Codepage 437 (e.g. MS-DOS):</td>
</tr>
<tr>
<td>43 3a 5c 72 65 87 75 2e 74 78 74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Counter-example: ambiguous file URI:</th>
</tr>
</thead>
<tbody>
<tr>
<td>File URI, in any ASCII-compatible document:</td>
</tr>
<tr>
<td>&quot;file:///%E3%81%A1&quot;</td>
</tr>
</tbody>
</table>

| Possible interpretations of the file name, depending on the
(unknown) encoding of the file system: |
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>o UTF-8:</td>
</tr>
<tr>
<td>&lt;HIRAGANA LETTER TI (U+3061)&gt;</td>
</tr>
</tbody>
</table>
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. Software using file URIs MUST NOT grant greater access than would be available for other file access methods.

Additionally, 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]. As such, the userinfo field of a file URI, if present, MUST NOT contain a password.

6. IANA Considerations

IANA maintains the registry of URI Schemes [BCP115] at http://www.iana.org/assignments/uri-schemes/. This document defines the following URI scheme, so the "Permanent URI Schemes" registry has been updated accordingly.

+------------+--------------------------+-----------+
| URI Scheme | Description              | Reference |
+------------+--------------------------+-----------+
| file       | Host-specific file names | RFC XXXX  |
+------------+--------------------------+-----------+

RFC Editor Note: Replace XXXX with this RFC’s reference.
7. Acknowledgements

This specification is derived from [RFC1738], [RFC3986], and [I-D.draft-hoffman-file-uri] (expired); the acknowledgements in those documents still apply.

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 comments and suggestions.

8. References

8.1. Normative References


8.2. Informative References


Appendix A. UNC Syntax

The UNC filesystem selector string is a null-terminated sequence of characters from the Universal Character Set [ISO10646].

The syntax of a UNC filesystem selector string, as defined by [MS-DTYP], is given here in Augmented Backus-Naur Form (ABNF) [RFC5234] for convenience:

```
UNC = "\\" hostname "\\" sharename *( "\\" objectname )
hostname = netbios-name / fqdn / ip-address
sharename = <name of share or resource to be accessed>
objectname = <depends on resource being accessed>
```

- "netbios-name" from [MS-NBTE], Section 2.2.1.
- "fqdn" from [RFC1035] or [RFC1123]
- "ip-address" from Section 2.1 of [RFC1123], or Section 2.2 of [RFC4291].

The precise format of "sharename" depends on the protocol; see: SMB [MS-SMB], NFS [RFC3530], NCP [NOVELL].

Author’s Address

Matthew Kerwin
QUT

Email: matthew.kerwin@qut.edu.au
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 invent new error response formats for HTTP APIs.

Note to Readers

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

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must
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-html401-19991224] 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 may 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": "http://example.com/probs/out-of-credit",
  "title": "You do not have enough credit.",
  "detail": "Your current balance is 30, but that costs 50.",
  "instance": "http://example.net/account/12345/msgs/abc",
  "balance": 30,
  "accounts": ["http://example.net/account/12345",
                "http://example.net/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.

3.1. Problem Details Object Members

A problem details object MAY have the following members:

- "type" (string) - An absolute URI [RFC3986] that identifies the problem type. When dereferenced, it SHOULD provide human-readable documentation for the problem type (e.g., using HTML [W3C.REC-html401-19991224]). 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.

- "status" (number) - The HTTP status code ([RFC7231], Section 6) generated by the origin server for this occurrence of the problem.
3. 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.

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. 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 may 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).

New problem type definitions MUST document:

1. A type URI (typically, with the "http" scheme),
2. A title that appropriately describes it (think short), and
3. The HTTP status code for it to be used with.

Problem types MAY specify the use of the Retry-After response header in appropriate circumstances.

A problem’s type URI SHOULD resolve to HTML [W3C.REC-html401-19991224] 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]) and SHOULD consist of characters from ALPHA, DIGIT, 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. 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; unrecognized parameters should be ignored
Encoding considerations: Same as [RFC7159]
Security considerations: see [this document]
Interoperability considerations: None.
Published specification: [this document]
Applications that use this media type: HTTP
Additional information:
    Magic number(s): n/a
    File extension(s): n/a
    Macintosh file type code(s): n/a
Person & 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; unrecognized parameters should be ignored
Encoding considerations: Same as [RFC7303]
Security considerations: see [this document]
Interoperability considerations: None.
Published specification: [this document]
Applications that use this media type: HTTP
Additional information:
    Magic number(s): n/a
    File extension(s): n/a
    Macintosh file type code(s): n/a
Person & 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

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


8.2. Informative References


Appendix A. HTTP Problems and XML

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

The OPTIONAL RELAX NG schema [ISO-19757-2] for the XML format is:

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 can be 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, an alternate version of the example above would appear in XML as:

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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<problem xmlns="urn:ietf:rfc:XXXX">
  <type>http://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>
    http://example.net/account/12345/msgs/abc
  </instance>
  <balance>30</balance>
  <accounts>
    <i>http://example.net/account/12345</i>
    <i>http://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.

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-html401-19991224]) 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:
or by inventing a mapping into RDFa [W3C.REC-rdfa-core-20120607].

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
UC Berkeley

Email: dret@berkeley.edu
URI: http://dret.net/netdret/
Message Disposition Notification
draft-ietf-appsawg-mdn-3798bis-02.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.

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 memo defines a RFC-MIME-MEDIA [4] content-type 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 [6].

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

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

This memo is currently marked with the ‘pre5378Trust200902’ IPR
statements until a release has been obtained from all previous
authors and editors of this text.

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 [9].

All syntax descriptions use the ABNF specified by RFC-MSGFMT [2], in which the lexical tokens (used below) are defined: "CRLF", "FWS", "CFWS", "field-name", "mailbox", "msg-id", and "text". The following lexical tokens are defined in RFC-SMTP [1]: "atom". The following lexical tokens are defined in the definition of the Content-Type header field in RFC-MIME-BODY [3]: "attribute" and "value".

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 MUA 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

```plaintext
mdn-request-header = "Disposition-Notification-To" "::" [FWS]
  mailbox *("," [FWS] mailbox)
```

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 [13], it SHOULD implement the procedures specified in RFC-IMAP-MDN [10].

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.

MDNs SHOULD 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 [2]). In this case, confirmation from the user SHOULD be obtained, if possible. If obtaining consent is not possible (e.g., because the user is not online at the time), then an MDN SHOULD NOT be sent.

Confirmation from the user SHOULD 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 should be done using only the addr-spec (local-part "@" domain) portion, excluding any angle brackets, phrase and route. The comparison MUST be 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 [2] for more details.) Implementations MAY treat known domain aliases as equivalent for the purpose of comparison.

Note that use of subaddressing (see [12]) 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. [[ more work needed here ]]

[[CREF1: (From Bruce) Of those, the angle bracket issue ought to be understood, but clarification could benefit implementors, especially as RFC 5322 defined the Return-Path syntax somewhat peculiarly. Canonicalization of local-parts and domains should probably be required prior to comparison, and use of on-the-wire forms should probably also be specified. DNS equivalence issues might be tricky]
for some implementations (e.g. offline reading); perhaps the specification could use RFC 2119 "MAY" to give implementations leeway to consider A vs. CNAME and DNS vs domain literal equivalence for situations where DNS is available to the implementation (I’m not sure about MX). About the only thing that can be said w.r.t. subaddressing and subdomains is a caution to sending MUA and address-rewriting MTA authors that a mismatch might result in no MDN being produced.

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.

A message that contains a Disposition-Notification-To header field SHOULD also contain a Message-ID header field as specified in RFC-MSGFMT [2]. This will permit automatic correlation of MDNs with their original messages by user agents.

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 for whom message disposition notifications are requested and for 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.

Messages posted to newsgroups SHOULD NOT have a Disposition-Notification-To header field.

2.2. The Disposition-Notification-Options Header

Future 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

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

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

importance = "required" / "optional"

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 beginning with "X-" will never
be defined as standard names; such names are reserved for
experimental use. Disposition-notification-parameter attribute names
not beginning with "X-" MUST be registered with the Internet Assigned
Numbers Authority (IANA) and described in a standards-track RFC or an
experimental RFC approved by the IESG. [[ more work needed here ]]
(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 MTA. 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 [1] and RFC-DSN-SMTP [7].

RFC-DSN-SMTP [7] 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
text of this header field is as follows:

original-recipient-header =
   "Original-Recipient" ":" [FWS] address-type [FWS] ";" [FWS] generic-address

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 Content 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 [4]) 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 [4]). These header fields SHOULD NOT be used in the header fields of any of the fragments themselves.

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 [4]), they pertain to an MDN that will be generated for the reassembled message. Section 5.2.2.1 of RFC-MIME-MEDIA [4]) 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 [6]). 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 [6].
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, encrypted message text MUST be returned, if it is returned at all, only 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 [2] 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 field of the message 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 or 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, MDNs may not be generated for some recipients for which MDNs were requested.
3.1. The message/disposition-notification content-type

The message/disposition-notification content-type is defined as follows:

MIME type name: message

MIME subtype name: disposition-notification

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 this memo.

(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 [2] 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
   *( failure-field CRLF )
   *( error-field CRLF )
   *( extension-field CRLF )
extension-field = extension-field-name "::" *(CFWS / text)
extension-field-name = field-name
```

Note that the order of the above fields is fixed, with the exception of the extension fields.
3.1.1. General conventions for fields

Since these fields are defined according to the rules of RFC-MSGFMT [2], 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. Comments in notification fields may use the "encoded-word" construct defined in RFC-MIME-HEADER [5].

3.1.2. "*-type" subfields

Several fields consist of a "*-type" subfield, followed by a semi-colon, followed by "*text". [[[ more work needed here ]]
[[CREF2: (Here and elsewhere in the document) It looks like there is emerging consensus that the document should describe 2 grammars: one for "must accept" (which allows CFWS) and another one for "should generate" (CFWS not allowed, but [FWS] are allowed). A future version of this document will implement this change, unless objections are voiced. ]]

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.

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.

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 [7].) Registration forms for address-type and mta-name-type appear in RFC-DSN-FORMAT [8].

3.2. Message/disposition-notification Fields

3.2.1. The Reporting-UA field

reporting-ua-field = "Reporting-UA" "::" ua-name [ "::" ua-product ]
ua-name = *text-no-semi
ua-product = *text-no-semi
text-no-semi = %d1-9 / %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. This field is optional, but recommended. For Internet Mail user agents, it is recommended that this field contain both: the DNS name of the particular instance of the MUA that generated the MDN, and the name of the product. For example,

Reporting-UA: pc.example.com; Foomail 97.1

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.

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" "::" mta-name-type "::" mta-name
mta-name = *text

For gateways into Internet Mail, the MTA-name-type will normally be "smtp", 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 no Original-Recipient header field in the message, then the Original-Recipient field MUST be omitted, unless the same information is reliably available some other way. If there is an Original-Recipient header field in the original message (or original recipient information is reliably available some other way), then the Original-Recipient field must be supplied. 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" "::" [FWS] address-type [FWS] "::" [FWS] generic-address

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 [2]. 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" "::" [FWS] address-type [FWS] "::" [FWS] generic-address

generic-address = *text

The generic-address subfield of the Final-Recipient field MUST contain the mailbox address of the recipient (from the From header field of the MDN) as it was when the MDN was generated by the MUA.

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".

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 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 [2].

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" ":" [FWS] disposition-mode ";"
  [FWS] disposition-type
  [";" disposition-modifier
  [";" disposition-modifier ] ]

disposition-mode = action-mode "/" [FWS] sending-mode
action-mode = "manual-action" / "automatic-action"
sending-mode = "MDN-sent-manually" / "MDN-sent-automatically"

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

disposition-modifier = [FWS]
  ("error" / disposition-modifier-extension)

disposition-modifier-extension = atom
The disposition-mode, disposition-type, and disposition-modifier may be spelled in any combination of upper and lower case characters.

3.2.6.1. Disposition modes

The following disposition 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.

"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.

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

"MDN-sent-manually" The user explicitly gave permission for this particular MDN to be sent.

"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. Disposition value names beginning with "X-" will never be defined as standard values; such names are reserved for experimental use. MDN disposition value names NOT beginning with "X-" MUST be registered with the Internet Assigned Numbers Authority (IANA) and described in a standards-track RFC or an experimental RFC approved by the IESG. (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.

If an MUA developer does not wish to register the meanings of such disposition modifier extensions, "X-" modifiers may be used for this purpose. To avoid name collisions, the name of the MUA implementation should follow the "X-", (e.g., "X-Foomail-").

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. Failure and Error Fields

The Failure and Error fields are used to supply additional information in the form of text messages when the "failure" disposition type or "error" disposition modifier appear. The syntax is as follows:

failure-field = "Failure" ":\" *text
error-field = "Error" ":\" *text

3.3. Extension-fields

Additional MDN fields may be defined in the future by later revisions or extensions to this specification. Extension-field names beginning with "X-" will never be defined as standard fields; such names are reserved for experimental use. MDN field names NOT beginning with "X-" MUST be registered with the Internet Assigned Numbers Authority (IANA) and described in a standards-track RFC or an experimental RFC approved by the IESG. (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).

If an application developer does not wish to register the meanings of such extension fields, "X-" fields may be used for this purpose. To avoid name collisions, the name of the application implementation should follow the "X-", (e.g., "X-Foomail-Log-ID" or "X-Foomail-EDI-info").
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 MTA (original recipient information passed along)

-- MTA sends message to next MTA

-- Final MTA receives message

-- Final MTA delivers message to MUA (possibly generating a DSN)

-- MUA performs automatic processing and generates corresponding MDNs ("dispatched", "processed" or "deleted" disposition type with "automatic-action" and "MDN-sent-automatically" disposition modes)

-- MUA displays list of messages to user

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

-- MUA performs requested action and, 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).

-- 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 [7] 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 [7], section 6.2.7.3), each of the recipients may issue an MDN.

Successful distribution of a message to a mailing list exploder 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, the mailing list exploder MAY 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 MAY 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 may be 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

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). 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 could reveal confidential information about host names and/or network topology inside a firewall.

An unencrypted MDN could reveal confidential information about an encrypted message, especially if all or part of the original message is returned in part 3 of the multipart/report. Encrypted MDNs are not defined in this specification.

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 along with a source route. The source route is ignored in 
the comparison so the addresses will always match. But if the source 
route is honored when the notification is sent, it could direct the 
message to some other destination. This risk can be minimized by not 
sending MDN’s automatically.

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-

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. Such an 
attack could overrun the capacity of the targeted mailbox and deny 
service.

For that reason, MDN’s SHOULD NOT be sent automatically where the 
"disposition-notification-to:" address is different from the envelope 
MAIL FROM address. See Section 2.1 for further discussion.

7. Collected Grammar

NOTE: The following lexical tokens are defined in RFC-MSGFMT [2]: 
CRLF, FWS, CFWS, field-name, mailbox, msg-id, text. The following 
lexical tokens are defined in RFC-SMTP [1]: atom. (Note that RFC-
MSGFMT [2] also defines "atom", but the version from RFC-SMTP [1] is 
more restrictive and this more restrictive version is used in this 
document) The definitions of attribute and value are as in the 
definition of the Content-Type header field in RFC-MIME-BODY [3].

Message header fields:
mdn-request-header =
  "Disposition-Notification-To" "::" [FWS]
  mailbox *("," [FWS] mailbox)
Disposition-Notification-Options =
  "Disposition-Notification-Options" "::" [FWS]
  disposition-notification-parameter-list
disposition-notification-parameter-list =
  disposition-notification-parameter
  *("," [FWS] disposition-notification-parameter)
disposition-notification-parameter =
  attribute "=" [FWS]
  importance "," [FWS] value *(";" [FWS] value)
importance = "required" / "optional"
original-recipient-header =
  "Original-Recipient" "::" [FWS] address-type [FWS]
  generic-address
Report content:
  disposition-notification-content =
    [ reporting-ua-field CRLF ]
    [ mdn-gateway-field CRLF ]
    [ original-recipient-field CRLF ]
    final-recipient-field CRLF
    disposition-field CRLF
    *( failure-field CRLF )
    *( error-field CRLF )
    *( extension-field CRLF )
address-type = atom
mta-name-type = atom
reporting-ua-field = "Reporting-UA" "::" ua-name [ ";" ua-product ]
uai-name = *text-no-semi
ua-product = *text-no-semi
text-no-semi = %[d1-9 / %[d11 / %[d12 / %[d14-58 / %[d60-127 LF, or semi-colon
mdn-gateway-field = "MDN-Gateway" "::" mta-name-type ";" mta-name
mta-name = *text
original-recipient-field =
  "Original-Recipient" "::" [FWS] address-type [FWS]
  generic-address
generic-address = *text
final-recipient-field =
  "Final-Recipient" "::" [FWS] address-type [FWS]
  generic-address
original-message-id-field = "Original-Message-ID" "::" msg-id
disposition-field =
  "Disposition" "::" [FWS] disposition-mode ";"
  [FWS] disposition-type
  *( "," disposition-modifier )
disposition-mode = action-mode "/" [FWS] sending-mode
action-mode = "manual-action" / "automatic-action"
sending-mode = "MDN-sent-manually" / "MDN-sent-automatically"
disposition-type = "displayed" / "deleted" / "dispatched" / "processed"
disposition-modifier = [FWS] ("error" / disposition-modifier-extension)
disposition-modifier-extension = atom
failure-field = "Failure" ":" *text
error-field = "Error" ":" *text
extension-field = extension-field-name ":" *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. Gateways 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. Gateways 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

This document specifies three types of parameters that must be registered with the Internet Assigned Numbers Authority (IANA).

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 standards track RFC or experimental RFC approved by the IESG 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 standards track RFC or experimental RFC approved by the IESG 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 standards track RFC or experimental RFC approved by the IESG that describes the semantics of the extension field.

11. Acknowledgements

The contributions of Bruce Lilly and Alfred Hoenes 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


12.2. Informative References


Appendix A. Changes from RFC 3798

The values of "dispatched" and "processed" were lost from the ABNF for "disposition-type".

Because the warning disposition modifier was previously removed, warning-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 and ua-product definitions were restricted to not include semi-colon.

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 [2]-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.

[[CREF3: Shouldn't the places we use *text and *text-no-semi allow FWS? ]]

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 [3] production of the same name, as well as other uses of the same term. These have been clarified.

[[CREF4: Not sure what to do with this one: (From Bruce) In the case of the message header fields, RFC 2822 also specifies minimum and maximum counts for each header field, and similar guidance would clarify 3798 (e.g. are multiple Disposition-Notification-Options fields permitted in a single message header, and if so, what semantics apply?). ]]  

[[CREF5: Not sure what to do with this one: (From Bruce) Note also that RFC 2045 is itself based on RFC 822 rather than 2822, so the issue of where CFWS is permitted or prohibited should probably be clearly specified where "attribute" and "value" are used. Note further that the RFC 2045 definitions are clarified by errata and modified by RFC 2231, and by RFC 2231 errata. Finally, note that RFC]
2231 has provisions for continuation of long parameter values (where there would otherwise be problems with the maximum line length specifications of RFCs 822 and 2822), specification of language and charset, and provision for compatible handling of non-ASCII text, none of which are provided for in the RFC 3798 disposition-notification parameters. It might be a good idea to think about that now, as a future change would almost certainly reset the document status to "Proposed".]

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. 
[[CREF6: Not sure what to do with this one: (From Bruce) 3.1.1 explicitly mentions use of RFC 2047 encoded-words in comments (however, as noted above there is no explicit provision for comments), but fails to mention the other contexts in which encoded-words may be used, viz. in an RFC [2]822 "phrase" (e.g. in the display name of a name-addr mailbox in Disposition-Notification-To (therefore, the discussion of encoded-words should probably be moved earlier in the document, prior to the specification of Disposition-Notification-To)), and in unstructured text (i.e. every instance of *text in the ABNF). In particular, use of encoded-words might be highly desirable in the following places: *) the ua-product portion of the Reporting-UA field; *) the generic-address part of the Original-Recipient and Final-Recipient fields; *) the (unstructured) field bodies of Error, Failure, and Warning fields; in structured extension fields where the context (per RFC 2047) is appropriate in unstructured extension fields; *) in X- extension fields (see RFC 2047 for related X- message header fields). In cases where the field syntax is shared with DSN fields, some coordination with the RFC 346x authors might be desirable. ]]

Authors' Addresses

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

Email: tony+rfc3798@mailennium.att.com
Abstract

This specification (re)defines the multipart/form-data Internet Media Type, which can be used by a wide variety of applications and transported by a wide variety of protocols as a way of returning a set of values as the result of a user filling out a form. It replaces RFC 2388.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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


1. Introduction

In many applications, it is possible for a user to be presented with a form. The user will fill out the form, including information that is typed, generated by user input, or included from files that the user has selected. When the form is filled out, the data from the form is sent from the user to the receiving application.

The definition of "multipart/form-data" is derived from one of those applications, originally set out in [RFC1867] and subsequently incorporated into HTML 3.2 [W3C.REC-html32-19970114], where forms are expressed in HTML, and in which the form data is sent via HTTP or
electronic mail. This representation is widely implemented in numerous web browsers and web servers.

However, "multipart/form-data" is also used for forms that are presented using representations other than HTML (spreadsheets, PDF, etc.), and for transport using means other than electronic mail or HTTP; it is used in distributed applications which do not involve forms at all, or do not have users filling out the form. For this reason, this document defines a general syntax and semantics independent of the application for which it is used, with specific rules for web applications noted in context.

2. percent-encoding option

Within this specification, "percent-encoding" (as defined in [RFC3986]) is offered as a possible way of encoding characters in file names that are otherwise disallowed, including non-ASCII characters, spaces, control characters and so forth. The encoding is created replacing each non-ASCII or disallowed character with a sequence, where each byte of the UTF-8 encoding of the character is represented by a percent-sign (%) followed by the (case-insensitive) hexadecimal of that byte.

3. Advice for Forms and Form Processing

The representation and interpretation of forms and the nature of form processing is not specified by this document. However, for forms and form-processing that result in generation of multipart/form-data, some suggestions are included.

In a form, there is generally a sequence of fields, where each field is expected to be supplied with a value, e.g. by a user who fills out the form. Each field has a name. After a form has been filled out, and the form’s data is "submitted": the form processing results in a set of values for each field-- the "form data".

In forms that work with multipart/form-data, field names could be arbitrary Unicode strings; however, restricting field names to ASCII will help avoid some interoperability issues (see Section 5.1).

Within a given form, insuring field names are unique is also helpful. Some fields may have default values or presupplied values in the form itself. Fields with presupplied values might be hidden or invisible; this allows using generic processing for form data from a variety of actual forms.
4. Definition of multipart/form-data

The media-type "multipart/form-data" follows the model of multipart MIME data streams as specified in [RFC2046] Section 5.1; changes are noted in this document.

A "multipart/form-data" body contains a series of parts, separated by a boundary.

4.1. Boundary parameter of multipart/form-data

As with other multipart types, the parts are delimited with a boundary delimiter, constructed using CRLF, "--", the value of the boundary parameter. The boundary is supplied as a "boundary" parameter to the "multipart/form-data" type. As noted in [RFC2046] Section 5.1, the boundary delimiter MUST NOT appear inside any of the encapsulated parts, and it is often necessary to enclose the boundary parameter values in quotes on the Content-type line.

4.2. Content-Disposition header for each part

Each part MUST contain a "content-disposition" header [RFC2183] and where the disposition type is "form-data". The "content-disposition" header MUST also contain an additional parameter of "name"; the value of the "name" parameter is the original field name from the form (possibly encoded; see Section 5.1). For example, a part might contain a header:

Content-Disposition: form-data; name="user"

with the body of the part containing the form data of the "user" field.

4.3. filename attribute of content-distribution part header

For form data that represents the content of a file, a name for the file SHOULD be supplied as well, by using a "filename" parameter of the "content-disposition" header. The file name isn't mandatory for cases where the file name isn't available or is meaningless or private; this might result, for example, from selection or drag-and-drop or where the form data content is streamed directly from a device.

If a filename parameter is supplied, the requirements of [RFC2183] Section 2.3 for "receiving MUA" apply to receivers of "multipart/form-data" as well: Do not use the file name blindly, check and possibly change to match local filesystem conventions if applicable, do not use directory path information that may seems to be present.
In most multipart types, the MIME headers in each part are restricted to US-ASCII; for compatibility with those systems, file names normally visible to users MAY be encoded using the percent-encoding method in Section 2, following how a "file:" URI [I-D.ietf-appsawg-file-scheme] might be encoded.

NOTE: The encoding method described in [RFC5987], which would add a "filename*" parameter to the "Content-Disposition" header, MUST NOT be used.

Some commonly deployed systems use multipart/form-data with file names directly encoded including octets outside the US-ASCII range. The encoding used for the file names is typically UTF-8, although HTML forms will use the charset associated with the form.

4.4. Multiple files for one form field

The form data for a form field might include multiple files.

[RFC2388] suggested that multiple files for a single form field be transmitted using a nested multipart/mixed part. This usage is deprecated.

To match widely deployed implementations, multiple files MUST be sent by supplying each file in a separate part, but all with the same "name" parameter.

Receiving applications intended for wide applicability (e.g. multipart/form-data parsing libraries) SHOULD also support the older method of supplying multiple files.

4.5. Content-Type header for each part

Each part MAY have an (optional) "content-type", which defaults to "text/plain". If the contents of a file are to be sent, the file data SHOULD be labeled with an appropriate media type, if known, or "application/octet-stream".

4.6. The charset parameter for text/plain form data

In the case where the form data is text, the charset parameter for the "text/plain" Content-Type MAY be used to indicate the character encoding used in that part. For example, a form with a text field in which a user typed "Joe owes &euro;100" where &euro; is the Euro symbol might have form data returned as:
Joe owes €100.

In practice, many widely deployed implementations do not supply a charset parameter in each part, but, rather, they rely on the notion of a "default charset" for a multipart/form-data instance. Subsequent sections will explain how the default charset is established.

4.7. The _charset_ field for default charset

Some form processing applications (including HTML) have the convention that the value of a form entry with entry name "_charset_" and type "hidden" is automatically set when the form is opened; the value is used as the default charset of text field values (see form-charset in Section 5.1.2). In such cases, the value of the default charset for each text/plain part without a charset parameter is the supplied value. For example:

```
--AaB03x
content-disposition: form-data; name="_charset_"
iso8859-1
--AaB03x--
content-disposition: form-data; name="field1"
...text encoded in iso-8859-1 ...
AaB03x--
```

4.8. Content-Transfer-Encoding deprecated

Previously, it was recommended that senders use a "Content-Transfer-Encoding" encoding (such as "quoted-printable") for each non-ASCII part of a multipart/form-data body, because that would allow use in transports that only support a "7BIT" encoding. This use is deprecated for use in contexts that support binary data such as HTTP. Senders SHOULD NOT generate any parts with a "Content-Transfer-Encoding" header.

Currently, no deployed implementations that send such bodies have been discovered.
4.9. Other Content-headers

The "multipart/form-data" media type does not support any MIME headers in the parts other than Content-Type, Content-Disposition, and (in limited circumstances) Content-Transfer-Encoding. Other headers MUST NOT be included and MUST be ignored.

5. Operability considerations

5.1. Non-ASCII field names and values

 Normally, MIME headers in multipart bodies are required to consist only of 7-bit data in the US-ASCII character set. While [RFC2388] suggested that non-ASCII field names be encoded according to the method in [RFC2047], this practice doesn’t seem to have been followed widely.

This specification makes three sets of recommendations for three different states of workflow.

5.1.1. Avoid non-ASCII field names

For broadest interoperability with existing deployed software, those creating forms SHOULD avoid non-ASCII field names. This should not be a burden, because in general the field names are not visible to users. The field names in the underlying need not match what the user sees on the screen.

If non-ASCII field names are unavoidable, form or application creators SHOULD use UTF-8 uniformly. This will minimize interoperability problems.

5.1.2. Interpreting forms and creating form-data

Some applications of this specification will supply a character encoding to be used for interpretation of the multipart/form-data body. In particular, HTML 5 [W3C.REC-html5-20141028] uses:

- The content of a ‘_charset_’ field, if there is one.
- the value of an accept-charset attribute of the <form> element, if there is one,
- the character encoding of the document containing the form, if it is US-ASCII compatible,
- otherwise UTF-8.
Call this value the form-charset. Any text, whether field name, field value, or (text/plain) form data which uses characters outside the ASCII range MAY be represented directly encoded in the form-charset.

5.1.3. Parsing and interpreting form data

While this specification provides guidance for creation of multipart/form-data, parsers and interpreters should be aware of the variety of implementations. File systems differ as to whether and how they normalize Unicode names, for example. The matching of form elements to form-data parts may rely on a fuzzier match. In particular, some multipart/form-data generators might have followed the previous advice of [RFC2388] and used the [RFC2047] "encoded-word" method of encoding non-ASCII values:

\[ \text{encoded-word} = "=?" \text{ charset } "?" \text{ encoding } "?" \text{ encoded-text } "?" \]

Others have been known to follow [RFC2231], to send unencoded UTF-8, or even strings encoded in the form-charset.

For this reason, interpreting "multipart/form-data" (even from conforming generators) may require knowing the charset used in form encoding, in cases where the _charset_ field value or a charset parameter of a text/plain Content-Type header is not supplied.

5.2. Ordered fields and duplicated field names

Form processors given forms with a well-defined ordering SHOULD send back results in order (note that there are some forms which do not define a natural order.) Intermediaries MUST NOT reorder the results. Form parts with identical field names MUST NOT be coalesced.

5.3. Interoperability with web applications

Many web applications use the "application/x-url-encoded" method for returning data from forms. This format is quite compact, e.g.:

\[ \text{name=Xavier+Xantico&verdict=Yes&colour=Blue&happy=sad&Utf%F6r=Send} \]

However, there is no opportunity to label the enclosed data with content type, apply a charset, or use other encoding mechanisms.

Many form-interpreting programs (primarily web browsers) now implement and generate multipart/form-data, but an existing application might need to optionally support both the application/x-url-encoded format as well.
5.4. Correlating form data with the original form

This specification provides no specific mechanism by which multipart/form-data can be associated with the form that caused it to be transmitted. This separation is intentional; many different forms might be used for transmitting the same data. In practice, applications may supply a specific form processing resource (in HTML, the ACTION attribute in a FORM tag) for each different form. Alternatively, data about the form might be encoded in a "hidden field" (a field which is part of the form but which has a fixed value to be transmitted back to the form-data processor.)

6. IANA Considerations

Please update the Internet Media Type registration of multipart/form-data to point to this document, using the template in Section 8. In addition, please update the registration of the "name" parameter in the "Content Disposition Parameters" registry to also point to this document.

7. Security Considerations

Applications which receive forms and process them must be careful not to supply data back to the requesting form processing site that was not intended to be sent.

It is important when interpreting the filename of the Content-Disposition header to not overwrite files in the recipient’s file space inadvertently.

User applications that request form information from users must be careful not to cause a user to send information to the requestor or a third party unwillingly or unwittingly. For example, a form might request ‘spam’ information to be sent to an unintended third party, or private information to be sent to someone that the user might not actually intend. While this is primarily an issue for the representation and interpretation of forms themselves (rather than the data representation of the form data), the transportation of private information must be done in a way that does not expose it to unwanted prying.

With the introduction of form-data that can reasonably send back the content of files from a user’s file space, the possibility arises that a user might be sent an automated script that fills out a form and then sends one of the user’s local files to another address. Thus, additional caution is required when executing automated scripting where form-data might include a user’s files.
Files sent via multipart/form-data may contain arbitrary executable content, and precautions against malicious content are necessary.

The considerations of [RFC2183] Sections 2.3 and 5 with respect to the filename parameter of the Content-Disposition header also apply to its usage here.

All form processing software should treat user supplied form-data with sensitivity, as it often contains confidential or personally identifying information. There is widespread use of form "auto-fill" features in web browsers; these might be used to trick users to unknowingly send confidential information when completing otherwise innocuous tasks. Multipart/form-data does not supply any features for checking integrity, ensuring confidentiality, avoiding user confusion, or other security features; those concerns must be addressed by the form-filling and form-data-interpreting applications.

8. Media type registration for multipart/form-data

This section is the [RFC6838] media type registration.

Type name: multipart

Subtype name: form-data

Required parameters: boundary

Optional parameters: none

Encoding considerations: Common use is BINARY.

In limited use (or transports that restrict the encoding to 7BIT or 8BIT each part is encoded separately using Content-Transfer-Encoding Section 4.8.

Security considerations: See Section 7 of this document.

Interoperability considerations: This document makes several recommendations for interoperability with deployed implementations, including Section 4.8.

Published specification: This document.

Applications that use this media type: Numerous web browsers, servers, and web applications.

Fragment identifier considerations: None: Fragment identifiers are not defined for this type.
Additional information: None: no deprecated alias names, magic numbers, file extensions or Macintosh sssssf type codes.

Person & email address to contact for further information
Author of this document.

Intended Usage: COMMON

Restrictions on usage: none

Author: Author of this document.

Change controller: IETF

Provisional registration: N/A

9. References

9.1. Normative References


9.2. Informative References

[I-D.ietf-appsawg-file-scheme]
Appendix A. Changes from RFC 2388

The handling of non-ASCII field names changed—no longer recommending the RFC 2047 method, instead suggesting senders send UTF-8 field names directly, and file names directly in the form-data charset.

The handling of multiple files submitted as the result of a single form field (e.g. HTML’s `<input type=file multiple>` element) results in each file having its own top level part with the same name parameter; the method of using a nested "multipart/mixed" from [RFC2388] is no longer recommended for creators, and not required for receivers as there are no known implementations of senders.

The _charset_ convention and use of an explicit form-data charset is documented.

‘boundary’ is a required parameter in Content-Type.

The relationship of the ordering of fields within a form and the ordering of returned values within multipart/form-data was not
defined before, nor was the handling of the case where a form has multiple fields with the same name.


Appendix B. Alternatives

There are numerous alternative ways in which form data can be encoded; many are listed in [RFC2388] section 5.2. The multipart/form-data encoding is verbose, especially if there are many fields with short values. In most use cases, this overhead isn’t significant.

More problematic is the ambiguity introduced because implementations did not follow [RFC2388] because it used "may" instead of "MUST" when specifying encoding of field names, and for other unknown reasons, so now, parsers need to be more complex for fuzzy matching against the possible outputs of various encoding methods.

Author’s Address

Larry Masinter
Adobe

Email: masinter@adobe.com
URI: http://larry.masinter.net
A "Null MX" No Service Resource Record for Domains that Accept No Mail
draft-ietf-appsawg-nullmx-10

Abstract

Internet mail determines the address of a receiving server through the DNS, first by looking for an MX record and then by looking for an A/AAAA record as a fallback. Unfortunately this means that the A/AAAA record is taken to be mail server address even when that address does not accept mail. The no service MX RR, informally called null MX, formalizes the existing mechanism by which a domain announces that it accepts no mail, without having to provide a mail server, which permits significant operational efficiencies.

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 March 17, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Conventions Used in This Document ..................... 2
2. Introduction ............................................. 2
3. MX Resource Records Specifying Null MX .................. 3
   4. Effects of Null MX .................................... 3
      4.1. SMTP Server Benefits ............................ 3
      4.2. Sending Mail from Domains that Publish Null MX ... 4
6. Security Considerations .................................. 5
7. Acknowledgements ........................................ 5
8. References .............................................. 5
   8.1. Normative References .............................. 5
   8.2. Informative References ............................. 6
Appendix A. Change Log .................................... 6
   A.1. Change to appsawg-nullmx-10 ....................... 6
   A.2. Change to appsawg-nullmx-09 ....................... 6
   A.3. Change to appsawg-nullmx-08 ....................... 6
   A.4. Change to appsawg-nullmx-07 ....................... 6
   A.5. Change to appsawg-nullmx-06 ....................... 7
   A.6. Change to appsawg-nullmx-05 ....................... 7
   A.7. Change to appsawg-nullmx-04 ....................... 7
   A.8. Change to appsawg-nullmx-03 ....................... 7
   A.9. Change to appsawg-nullmx-02 ....................... 7
   A.10. Change to appsawg-nullmx-1 ....................... 7
   A.11. Change to appsawg-nullmx-0 ....................... 7
8. Authors’ Addresses ..................................... 7

1. Conventions Used in This Document

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

The terms RFC5321.MailFrom and RFC5322.From are used as defined in [RFC5598].

2. Introduction

This document defines the No Service MX, informally called null MX, as a simple mechanism by which a domain can indicate that it does not accept email.
SMTP clients have a prescribed sequence for identifying a server that accepts email for a domain. Section 5 of [RFC5321] covers this in detail, but in essence the SMTP client first looks up a DNS MX RR and if that is not found it falls back to looking up a DNS A or AAAA RR. Hence this overloads an email service semantic onto a DNS record with a different primary mission.

If a domain has no MX records, senders will attempt to deliver mail to the hosts at the domain’s A or AAAA record’s addresses. If there is no SMTP listener at the A/AAAA address, message delivery will be attempted repeatedly for a long period, typically a week, before the sending MTA gives up. This will delay notification to the sender in the case of misdirected mail, and will consume resources at the sender.

This document defines a null MX that will cause all mail delivery attempts to a domain to fail immediately, without requiring domains to create SMTP listeners dedicated to preventing delivery attempts.

3. MX Resource Records Specifying Null MX

To indicate that a domain does not accept email, it advertises a single MX RR (see [RFC1035], section 3.3.9) with an RDATA section consisting of preference number 0, and a zero length label, written in master files as ".", as the exchange domain, to denote that there exists no mail exchanger for a domain. Since "." is not a valid host name, a null MX record can not be confused with an ordinary MX record. The use of "." as a pseudo-host name meaning no service available is modeled on the SRV RR [RFC2782] where it has a similar meaning.

A domain that advertises a null MX MUST NOT advertise any other MX RR.

4. Effects of Null MX

The null MX record has a variety of efficiency and usability benefits.

4.1. SMTP Server Benefits

Mail often has an incorrect address due to user error, where the address was mistranscribed or misunderstood, for example, to alice@www.example.com or alice@example.org or alice@example.com rather than alice@example.com. Null MX allows a mail system to report the delivery failure when the user sends the message, rather than hours or days later.
Senders of abusive mail often use forged undeliverable return addresses. Null MX allows Delivery Status Notifications (DSNs) and other attempted responses to such mail to be disposed of efficiently.

The ability to detect domains that do not accept email offers resource savings to an SMTP client. It will discover on the first sending attempt that an address is not deliverable, avoiding queuing and retries.

When a submission or SMTP relay server rejects an envelope recipient due to a domain’s null MX record, it SHOULD use a 556 reply code (Requested action not taken: domain does not accept mail) and a 5.1.TBD enhanced status code (Permanent failure: Recipient address has null MX).

A receiving SMTP server that chooses to reject email during the SMTP conversation that presents an undeliverable RFC5321.MailFrom or RFC5322.From domain can be more confident that for other messages a subsequent attempt to send a DSN or other response will reach a recipient SMTP server.

SMTP servers that reject mail because a RFC5321.MailFrom or RFC5322.From domain has a null MX record SHOULD use a 550 reply code (Requested action not taken: mailbox unavailable) and a 5.7.TBD enhanced status code (Permanent failure: Sender address has null MX).

4.2. Sending Mail from Domains that Publish Null MX

Null MX is primarily intended for domains that do not send or receive any mail, but have mail sent to them anyway due to mistakes or malice. Many receiving systems reject mail that has an invalid return address. Return addresses are needed to allow the sender to handle message delivery errors. An invalid return address often signals that the message is spam. Hence mail systems SHOULD NOT publish a null MX record for domains that they use in RFC5321.MailFrom or RFC5322.From addresses. If a server nonetheless does so, it risks having its mail rejected.

Operators of domains that do not send mail can publish SPF -all [RFC7208] policies to make an explicit declaration that the domains send no mail.

Null MX is not intended to be a replacement for the null reverse path described in RFC 5321 section 4.5.5 and does not change the meaning or use of a null reverse path.
5. Security Considerations

Within the DNS, a null MX RR is an ordinary MX record and presents no new security issues. If desired, it can be secured in the same manner as any other DNS record using DNSSEC.

6. IANA Considerations

IANA is requested to add the following entries to the "Enumerated Status Codes" sub-registry of the Simple Mail Transfer Protocol (SMTP) Enhanced Status Codes Registry.

Code: X.1.TBD
Sample Text: Recipient address has null MX
Associated basic status code: 556
Description: This status code is returned when the associated address is marked as invalid using a null MX.
Reference: [this document]
Submitter: [authors of this document]
Change controller: IESG

Code: X.7.TBD
Sample Text: Sender address has null MX
Associated basic status code: 550
Description: This status code is returned when the associated sender address has a null MX, and the SMTP receiver is configured to reject mail from such sender (e.g. because it could not return a DSN).
Reference: [this document]
Submitter: [authors of this document]
Change controller: IESG

7. Acknowledgements

We thank Dave Crocker for his diligent and lengthy shepherding of this document, and members of the appsawg working group for their constructive suggestions.

8. References

8.1. Normative References


8.2. Informative References


Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", RFC 7208, April 2014.

Appendix A. Change Log

*NOTE TO RFC EDITOR: This section may be removed upon publication of this document as an RFC.*

A.1. Change to appsawg-nullmx-10

Minor twiddle to clarify reference.

A.2. Change to appsawg-nullmx-09

Change 521 to 556, change reference.

A.3. Change to appsawg-nullmx-08

Fix name of IANA registry.

Yea, even yet more editorial cleanup.

A.4. Change to appsawg-nullmx-07

Add new enhanced status codes and ref for 521 return code.

Even yet more editorial cleanup.
A.5. Change to appsawg-nullmx-06

Even more editorial cleanup.

Mention SRV

you SHOULD NOT put a null MX on domains that send mail

A.6. Change to appsawg-nullmx-05

Fix ID nits, add NULL IANA section. More editorial cleanup.

A.7. Change to appsawg-nullmx-04

Reorganize.

A.8. Change to appsawg-nullmx-03

Editorial nits per Murray.

A.9. Change to appsawg-nullmx-02

Should not publish NULL MX with other MX.

Never say never.

Add 5.1.2 enhanced status code.

Minor editorial changes.

A.10. Change to appsawg-nullmx-1

Editorial improvements per D. Crocker’s review.

A.11. Change to appsawg-nullmx-0

Fix typos.

Authors’ Addresses

John Levine
Taughannock Networks
PO Box 727
Trumansburg, NY 14886

Phone: +1 831 480 2300
Email: standards@taugh.com
URI: http://jl.ly
Mark Delany
Apple Inc.
1 Infinite Loop
Cupertino, CA  95014

Email: mx0dot@yahoo.com
Message Header Field for Indicating Message Authentication Status
draft-ietf-appsawg-rfc7001bis-05

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 September 27, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction ................................................. 4
   1.1. Purpose .................................................. 5
   1.2. Trust Boundary ........................................... 5
   1.3. Processing Scope ........................................ 6
   1.4. Requirements ............................................ 6
   1.5. Definitions ............................................. 6
      1.5.1. Key Words ........................................... 7
      1.5.2. Security ............................................ 7
      1.5.3. Email Architecture ................................. 7
      1.5.4. Other Terms ........................................ 8
   1.6. Trust Environment ....................................... 9
2. Definition and Format of the Header Field ......................... 9
   2.1. General Description ..................................... 9
   2.2. Formal Definition ...................................... 10
   2.3. Property Types (ptypes) and Properties .................. 12
   2.4. The "policy" ptype ..................................... 13
   2.5. Authentication Identifier Field ........................ 14
   2.6. Version Tokens ......................................... 15
   2.7. Defined Methods and Result Values ....................... 15
      2.7.1. DKIM and DomainKeys ................................ 16
      2.7.2. SPF and Sender ID .................................. 17
      2.7.3. "iprev" ........................................... 18
      2.7.4. SMTP AUTH ........................................ 19
      2.7.5. Other Registered Codes ............................. 20
      2.7.6. Extension Methods ................................ 21
      2.7.7. Extension Result Codes ............................. 21
3. The "iprev" Authentication Method ................................ 22
4. Adding the Header Field to a Message .......................... 23
   4.1. Header Field Position and Interpretation ................ 24
   4.2. Local Policy Enforcement ................................ 25
5. Removing Existing Header Fields ................................. 26
6. IANA Considerations .......................................... 27
   6.1. The Authentication-Results Header Field ................ 27
   6.2. "Email Authentication Methods" Registry Description .... 27
   6.3. "Email Authentication Methods" Registry Update .......... 28
   6.4. "Email Authentication Property Types" Registry .......... 30
   6.5. "Email Authentication Result Names" Description .......... 30
   6.6. "Email Authentication Result Names" Update ............... 31
7. Security Considerations ...................................... 32
   7.1. Forged Header Fields .................................... 32
   7.2. Misleading Results ..................................... 34
   7.3. Header Field Position ................................... 34
7.4. Reverse IP Query Denial-of-Service Attacks ............... 34
7.5. Mitigation of Backscatter .................................. 34
7.6. Internal MTA Lists ........................................... 34
7.7. Attacks against Authentication Methods ...................... 35
7.8. Intentionally Malformed Header Fields ....................... 35
7.9. Compromised Internal Hosts .................................. 35
7.10. Encapsulated Instances ...................................... 35
7.11. Reverse Mapping .............................................. 36
8. References ................................................... 36
8.1. Normative References ......................................... 36
8.2. Informative References ....................................... 36
Appendix A. Acknowledgments ...................................... 39
Appendix B. Legacy MUAs .......................................... 39
Appendix C. Authentication-Results Examples ...................... 39
C.1. Trivial Case; Header Field Not Present ...................... 40
C.2. Nearly Trivial Case; Service Provided, but No Authentication Done .................. 40
C.3. Service Provided, Authentication Done ....................... 41
C.4. Service Provided, Several Authentications Done, Single MTA ...................................... 42
C.5. Service Provided, Several Authentications Done, Different MTAs .................. 43
C.6. Service Provided, Multi-Tiered Authentication Done ........ 45
C.7. Comment-Heavy Example ....................................... 46
Appendix D. Operational Considerations about Message Authentication ...................................... 47
Appendix E. Change History ....................................... 48
E.1. RFC7001 to -00 ............................................. 48
E.2. -00 to -01 ................................................ 49
E.3. -01 to -02 ................................................ 49
E.4. -02 to -03 ................................................ 49
E.5. -03 to -04 ................................................ 49
E.6. -04 to -05 ................................................ 50
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-readable 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, domain-level email authentication methods in common use:

- Author Domain Signing Practices ([ADSP])
- SMTP Service Extension for Authentication ([AUTH])
- DomainKeys Identified Mail Signatures ([DKIM])
- Sender Policy Framework ([SPF])
- Vouch By Reference ([VBR])
- reverse IP address name validation ("iprev", defined in Section 3)

In addition, the following are non-standard methods recognized by this specification that are no longer common:

- DomainKeys ([DOMAINKEYS]) (Historic)
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

- 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.)
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.

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.

```
+-----+   +-----+   +------------+
| MUA |-->| MSA |-->| Border MTA |
+-----+   +-----+   +------------+
  | V
  +--------+
  | Internet |
  +--------+
        | V
        +-----+   +-----+   +------------------+   +------------+
        | MUA |<--| MDA |<--| Intermediate MTA |<--| Border MTA |
        +-----+   +-----+   +------------------+   +------------+
```

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 / l*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

        ; 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
```
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
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, or
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:
o Changes to the identifier impose a large, centralized administrative burden.

o 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
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 is 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:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>[RFC7208], Section 2.6.1</td>
</tr>
<tr>
<td>pass</td>
<td>[RFC7208], Section 2.6.3</td>
</tr>
<tr>
<td>fail</td>
<td>[RFC7208], Section 2.6.4</td>
</tr>
<tr>
<td>softfail</td>
<td>[RFC7208], Section 2.6.5</td>
</tr>
<tr>
<td>policy</td>
<td>[this RFC], Section 2.4</td>
</tr>
<tr>
<td>neutral</td>
<td>[RFC7208], Section 2.6.2</td>
</tr>
<tr>
<td>temperror</td>
<td>[RFC7208], Section 2.6.6</td>
</tr>
<tr>
<td>permerror</td>
<td>[RFC7208], Section 2.6.7</td>
</tr>
</tbody>
</table>
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.

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.

temerror: 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, and 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, yet continued to send the message about which a result is being reported.
tempererror: 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

An agent making use of the data provided by this header field SHOULD consider "fail" and "tempererror" to be synonymous in terms of message authentication, i.e., the client did not authenticate in either case.

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");
o Require-Recipient-Valid-Since (in [RRVS], represented by "rrvs");

o 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.

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:
   Requesting review of any proposed changes and additions to this field is recommended.

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;

Defined: 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 "Defined" 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 in turn would 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 current entry for the "auth" method shall have its "property" field changed to "mailfrom", and its "Defined" field changed to this document.

2. 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.

3. All other "dkim", "domainkeys", "iprev", "sender-id", and "spf" method entries shall have their "Defined" fields changed to this document.

4. All "smime" entries have their "Defined" fields changed to [SMIME-REG].

5. The "value" field of the "smime" entry using property "smime-part" shall be changed to read "A reference to the MIME body part that contains the signature." The redundant reference is thus removed.

6. The following entry is to be added:

   Method: auth
   Defined: [this document]
   ptype: smtp
   property: auth
   Value: identity confirmed by the AUTH command
   Status: active
   Version: 1

7. 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 if not authenticated

   sender: contents of the [MAIL] Sender: header field, after removing comments, and removing the local-part if not authenticated
8. 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.

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.

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.

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.

6.6. "Email Authentication Result Names" Update

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 changed to refer to this document, as follows:

  dkim: [this document] Section 2.7.1
  domainkeys: [this document] Section 2.7.1
  spf: [this document] Section 2.7.2
  sender-id: [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 "dkim-adsp" and "domainkeys" shall have their Status values changed to "deprecated", reflecting the fact that the corresponding specifications now have Historical status.

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


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


8.2. Informative References


[AR-VBR] Kucherawy, M., "Authentication-Results


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 ESMTP id g1G0rlkA003489;
    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 i7PK0shH7021929;
           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:
                b=sEuZGD/pSr7ANysbY3jtdaQ3Xv9xPQtSOm70;
                b=ETbRSuvUFQVP3Bkz ... 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 ESMTP id glG0r1kA003489;
           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:

```plaintext
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=ftA9J6GtX8OpwUECzHnCkRzKw1uk6FNiIfj15Nmv49E=;
b=oineO8hgn/gunsg ... 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=Eu28nfs9fuZGD/pSr7ANysbY3jtadq3Xv9xPQtS0m7=;
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 History

E.1. RFC7001 to -00

- Remove "Changes since RFC5451" section; add this "Change History" section.
- Restore XML to previous format. (No visible changes).
- Reset "Acknowledgments".
- Add "To-Do" section.
E.2. -00 to -01

- Apply RFC7410.
- Update all the RFC4408 references to RFC7208.
- Add section explaining "property" values. (Errata #4201)
- Remove "To-Do" section.

E.3. -01 to -02

- Consolidate new sections.

E.4. -02 to -03

- Move the DKIM exception text down to where the DKIM results are defined, and add a forward reference to them.
- More detail about registry creation and previous augmentations.
- 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.
- Add references: [DMARC], [PRA], [RFC6577], [RRVS], [SMIME-REG].

E.5. -03 to -04

- Add specific update instructions for the "dkim"/"header"/"b" entry in IANA Considerations.
- Add description of values that can be extracted from SMTP AUTH sessions and an example.
- Much more complete descriptions of reporting DomainKeys results.
- Minor editorial adjustments.
- Fix up "smime" entries.
- Update current definitions for the Email Authentication Property Types registry to point to this document.
- Rework the Email Authentication Result Names registry.
- Add more detail about Sender ID.
- Mark all ADSP and DomainKeys entries as deprecated since their defining documents are as well.
- Add references: [RFC6008].

E.6. -04 to -05
- Fix typos.
- 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.
- Move the RFC6008 reference up to where the DKIM reporting is described.

Author’s Address

Murray S. Kucherawy
270 Upland Drive
San Francisco, CA 94127
US
EMail: superuser@gmail.com
The text/markdown Media Type
draft-ietf-appsawg-text-markdown-06

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

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
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: a linear sequence of characters in some character set (code), possibly interrupted by line breaks, page breaks, or other control characters. The repertoire of these control characters (a form of in-band signaling) is necessarily limited, and not particularly extensible. Because they are non-printing, these characters are also hard to enter with standard keyboards.

Markup offers an alternative means to encode this signaling information by overloading certain characters with additional meanings. Therefore, markup languages allow for annotating a document in such a way that annotations are syntactically distinguishable from the printing information. 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 author's intent on how to interpret the Markdown. 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 [MDMTUSES] provides additional Markdown background and philosophy.

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:

<table>
<thead>
<tr>
<th>File</th>
<th>Edit</th>
<th>(Cloud Stuff)</th>
<th>(Fork Me on GitHub)</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(plain text, with syntax highlighting)</th>
<th>(text/html, likely rendered to screen)</th>
</tr>
</thead>
<tbody>
<tr>
<td># Introduction</td>
<td>&lt;h1&gt;Introduction&lt;/h1&gt;</td>
</tr>
<tr>
<td>## Markdown Is About Writing and Editing</td>
<td>&lt;h2&gt;Markdown Is About Writing and Editing&lt;/h2&gt;</td>
</tr>
<tr>
<td>&gt; HTML is a <em>publishing</em> format;</td>
<td>&lt;p&gt;HTML is a &lt;em&gt;publishing&lt;/em&gt; format;&lt;/p&gt;</td>
</tr>
<tr>
<td>&gt; Markdown is a <em>writing</em> format.</td>
<td>&lt;em&gt;Markdown is a &lt;em&gt;writing&lt;/em&gt; format.&lt;/em&gt;</td>
</tr>
</tbody>
</table>
| > Thus, Markdown’s formatting syntax only addresses issues that can be conveyed in plain text. [MDSYNTAX][] | <<formatting syntax only addresses issues that can be conveyed in plain text. <a href="http://daringfireball.net/projects/markdown/syntax#html" title="Markdown: Syntax: HTML">MDSYNTAX</a>" >><p>The paradigmatic use case for "text/markdown" is the Markdown editor: an application that presents Markdown content ...

[MDSYNTAX]: http://daringfireball.net/projects/markdown/syntax#html
"Markdown: Syntax: HTML"

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

Users on diverse platforms SHOULD be able to collaborate with their tools of choice, whether those tools are desktop-based (MarkdownPad, MultiMarkdown Composer), browser-based (Dillinger, Markable), integrated widgets (Discourse, GitHub), general-purpose editors (emacs, vi), or plain old "Notepad". Additionally, users SHOULD be able to identify particular areas of Markdown content when the Markdown becomes appreciably large (e.g., book chapters and Internet-Drafts--not just blog posts). Users SHOULD be able to use text/markdown 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). 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. [MDSYNTAX] clearly describes Markdown as a writing format; its syntax rules operate on characters (specifically, on punctuation) rather than code points. Neither [MDSYNTAX] nor many popular implementations at the time of this registration actually require or assume any particular character set. Many Markdown processors will get along just fine by operating on character codes that lie in printable US-ASCII, blissfully oblivious to coded values outside of that range.

Optional parameters:

variant: An optional identifier that serves as a "hint" to the recipient of the specific Markdown variant that the author intended. 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 MAY 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 parameters. Additionally, the variant MAY be registered under another media type; this text/markdown registration does not preclude other registrations.

Encoding considerations: Text.

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, proportional to 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.

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 4.

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 [MDMTUSES] 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 [MDMTUSES].

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 output-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 in the Standards tree 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.

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
- Name
- Description
- Additional Parameters (optional)
- Fragment Identifiers (optional)
- References
- Contact Information
- Expiration Date (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*( [-" / "." / ":" / "~" ] 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.

Leonard Exp. August 27, 2015 [Page 10]
The Name, Description, Additional Parameters, Fragment Identifiers, References, and Contact Information fields SHALL be in a Unicode character set (e.g., UTF-8).

6.2. Reserved Identifiers

The registry SHALL have the following identifiers RESERVED. No one is allowed to register them (or any case variations of them).

- Standard
- Common
- Markdown

6.3. 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.

If a registration is being updated, the contact information MUST either match the prior registration and be verified, or the prior registrant MUST confirm that the updating registrant has authority to update 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.

6.4. 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.

7. Security Considerations

See the Security considerations entry in Section 2.

8. References

8.1. Normative References


8.2. Informative References


Appendix A. Change Log

This draft is a continuation from draft-ietf-appsawg-text-markdown-05.txt. These technical changes were made:

1. Removed TODO items for the time being.
3. Made minor changes.

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/

Leonard            Exp. August 27, 2015
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."

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
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: a linear sequence of characters in some character set (code), possibly interrupted by line breaks, page breaks, or other control characters. Plain text provides some fixed facilities for formatting instructions, namely codes in the character set that have meanings other than "represent this character 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 format: a sequence of instructions 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 byte or bit are 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 format 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-TAHN SIGN"; it means the start of a tag or element that affects the document in some way.

On the formal end of the 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 [CITE], for screenplays [FOUNTAIN], for mathematical formulae [CITE], and even for music annotation [CITE]. 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.

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] (draft-05) only defines two parameters: the charset parameter (required for all text/* media types) and the variant parameter. Character set interoperability is well-studied territory [NB: CITE?] and so is not further covered here. 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".
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

This strategy is to process the Markdown into the formal markup, which eliminates ambiguities. Once the Markdown is processed into (for example) valid XHTML, an application can save a file as "doc.xhtml" 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. The filename SHOULD be consistent with this Content-Type header, i.e., the extension SHOULD be the variant identifier plus ".markdown".

TODO: Versions of Subversion after [[1.x]] treat svn:mime-type as UTF-8 encoded, rather than US-ASCII. (See [RFC6532].) Therefore, the encoding of [RFC2231] will not be necessary in the vast majority of cases in newer versions. However, both for backwards compatibility and for support for non-Unicode character sets, [RFC2231] still needs to be supported.]]

TODO: Where to store Content-Disposition?

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. The encoding of the file should be transcoded to UTF-8. For other properties, a header file should be recorded alongside the Markdown file in accordance with Section 2.2. The contents of the header file should be consistent with the rest of this paragraph, i.e., the charset parameter should be "UTF-8" and the variant parameter should match the identifier in the filename.

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>
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:

[[Stuff to turn off:]]

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

[[New stuff:]]

lists_without_preceding_blankline
hard_line_breaks
ignore_line_breaks
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.

```markdown
#/       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
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

# MultiMarkdown 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. ""Oops this is some mistaken text.""

```javascript
function test() {
  return "notice this feature?";
}
```

# Table Alignments #

<table>
<thead>
<tr>
<th>Left</th>
<th>Center</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>cats</td>
<td>Paxton</td>
<td>$1600</td>
</tr>
<tr>
<td>dogs</td>
<td>Ruff</td>
<td>$30</td>
</tr>
<tr>
<td>zebras</td>
<td>Stripes</td>
<td>$20900</td>
</tr>
</tbody>
</table>

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

My other header

---

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: `\&nbsp;` & `\&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:**

RFCC2119:

**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—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.
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.

Sample Implementation

```php
<?php
header("Cache-Control: no-cache, must-revalidate");
header("Expires: Sat, 26 Jul 1997 05:00:00 GMT");
header("Content-Type: application/octet-stream");
?>
```

Figure 6: Markdown for RFCs Example
4.7. rfc7328 (Pandoc2rfc)

Pandoc2rfc expects multiple files as input. The following figure is an example of "middle.mkd".

# Introduction

```xml
toc="yes"
symrefs="yes"
sortrefs="yes"
subcompact="no"
compact="yes"
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].


# 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


7.2. Informative References


[MULTIMD] Penney, F., "MultiMarkdown", April 2014,
<http://fletcherpenney.net/multimarkdown/>.


Leonard                 Exp. September 10, 2015                [Page 24]
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/
Guidelines and Registration Procedures for URI Schemes
draft-ietf-appsawg-uri-scheme-reg-04

Abstract

This document updates the guidelines and recommendations, as well as the IANA registration processes, for the definition of Uniform Resource Identifier (URI) schemes. It obsoletes RFC 4395.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 23, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

The Uniform Resource Identifier (URI) protocol element and generic syntax is defined by [RFC3986]. Each URI begins with a scheme name, as defined by Section 3.1 of RFC 3986, that refers to a specification for identifiers within that scheme. The URI syntax provides a federated and extensible naming system, where each scheme's specification can further restrict the syntax and define the semantics of identifiers using that scheme.

This document obsoletes [RFC4395], which in turn obsoleted [RFC2717] and [RFC2718]. Recent documents have used the term "URI" for all...
resource identifiers, avoiding the term "URL" and reserving the term "URN" explicitly for those URIs using the "urn" scheme name ([RFC2141]). URN "namespaces" ([RFC3406]) are specific to the "urn" scheme and are not covered explicitly by this specification.

This document provides updated guidelines for the definition of new schemes, for consideration by those who are defining, registering, or evaluating those definitions, as well as a process and mechanism for registering schemes within the IANA URI Schemes registry. There is a single namespace for registered schemes. The intent of the registry is to:

- provide a central point of discovery for established URI scheme names, and easy location of defining documents for standard schemes;
- discourage multiple separate uses of the same scheme name;
- help those proposing new scheme names to discern established trends and conventions, and avoid names that might be confused with existing ones;
- encourage registration by setting a low barrier for registration.

As originally defined, URIs only allowed a limited repertoire of characters chosen from US-ASCII. An Interationalized Resource Identifier (IRI), as defined by [RFC3987], extends the URI syntax to allow characters from a much greater repertoire, to accomodate resource identifiers from the world’s languages. RFC 3987 [RFC3987] also defined a mapping between URIs and IRIs. A URI scheme name is the same as the corresponding IRI scheme name. Thus, there is no separate, independent registry or registration process for IRI schemes: the URI Schemes registry is used for both URIs and IRIs. Those who wish to describe resource identifiers that are useful as IRIs should define the corresponding URI syntax, and note that the IRI usage follows the rules and transformations defined in [RFC3987].

[RFC3986] defines the overall syntax for URIs as:

\[ URI = scheme ":" hier-part [ "?" query ] [ "#" fragment ] \]

A scheme definition cannot override the overall syntax for URIs. For example, this means that fragment identifiers(#) cannot be re-used outside the generic syntax restrictions. A scheme definition must specify the scheme name and the syntax of the scheme-specific part, which is clarified as follows:
URI = scheme ":" scheme-specific-part [ "#" fragment ]
scheme-specific-part = hier-part [ "?" query ]

2. Terminology

Within this document, the key words MUST, MAY, SHOULD, REQUIRED, RECOMMENDED, and so forth are used within the general meanings established in [RFC2119], within the context that they are requirements on future registrations.

This document distinguishes between a "scheme specification", being a document defining the syntax and semantics of a scheme, vs. a "scheme registration request" being the request submitted to IANA. The term "scheme definition" refers generically to the syntax and semantics of a scheme, typically documented in a scheme specification.

3. Requirements for Permanent Scheme Definitions

This section gives considerations for new schemes. Meeting these guidelines is REQUIRED for permanent scheme registration. Permanent status is appropriate for, but not limited to, use in standards. For IETF Standards-Track documents, Permanent registration status is REQUIRED.

3.1. Demonstrable, New, Long-Lived Utility

In general, the use and deployment of new schemes in the Internet infrastructure can be costly; some parts of URI processing are often scheme-dependent. Introducing a new scheme might require additional software, not only for client software and user agents but also in additional parts of the network infrastructure (gateways, proxies, caches) [W3CWebArch]. Since scheme names share a single, global namespace, it is desirable to avoid contention over use of short, mnemonic scheme names. New schemes ought to have utility to the Internet community beyond that available with already registered schemes. The scheme specification SHOULD discuss the utility of the scheme being registered.

3.2. Syntactic Compatibility

[RFC3986] defines the generic syntax for all URI schemes, along with the syntax of common URI components that are used by many URI schemes to define hierarchical identifiers. [RFC3987] extended this generic syntax to cover IRIs. All scheme specifications MUST define their own URI <scheme-specific-part> syntax. Care must be taken to ensure
that all strings matching their scheme-specific syntax will also
match the <absolute-URI> grammar described in [RFC3986].

New schemes SHOULD reuse the common URI components of [RFC3986] for
the definition of hierarchical naming schemes. If there is a strong
reason for a scheme not to use the hierarchical syntax, then the new
scheme definition SHOULD follow the syntax of previously registered
schemes.

Schemes that are not intended for use with relative URIs SHOULD avoid
use of the forward slash "/" character, which is used for
hierarchical delimiters, and the complete path segments "." and ".."
(dot-segments).

Schemes SHOULD avoid improper use of "//". The use of double slashes
in the first part of a URI is not a stylistic indicator that what
follows is a URI: Double slashes are intended for use ONLY when the
syntax of the <scheme-specific-part> contains a hierarchical
structure. In URIs from such schemes, the use of double slashes
indicates that what follows is the top hierarchical element for a
naming authority. (Section 3.2 of RFC 3986 has more details.)
Schemes that do not contain a conformant hierarchical structure in
their <scheme-specific-part> SHOULD NOT use double slashes following
the "<scheme>:" string.

New schemes SHOULD clearly define the role of [RFC3986] reserved
caracters in URIs of the scheme being defined. The syntax of the
new scheme should be clear about which of the "reserved" set of
caracters are used as delimiters within the URIs of the new scheme,
and when those characters must be escaped, versus when they can be
used without escaping.

3.3. Well-Defined

While URIs might or might not be defined as locators in practice, a
scheme definition itself MUST be clear as to how it is expected to
function. Schemes that are not intended to be used as locators
SHOULD describe how the resource identified can be determined or
accessed by software that obtains a URI of that scheme.

For schemes that function as locators, it is important that the
mechanism of resource location be clearly defined. This might mean
different things depending on the nature of the scheme.

In many cases, new schemes are defined as ways to translate between
other namespaces or protocols and the general framework of URIs. For
example, the "ftp" scheme translates into the FTP protocol, while the
"mid" scheme translates into a Message-ID identifier of an email
message. For such schemes, the description of the mapping MUST be complete, and in sufficient detail so that the mapping in both directions is clear: how to map from a URI into an identifier or set of protocol actions or name in the target namespace, and how legal values in the base namespace, or legal protocol interactions, might be represented in a valid URI. See Section 3.6 for guidelines for encoding binary or character strings within valid character sequences in a URI. If not all legal values or protocol interactions of the base standard can be represented using the scheme, the definition SHOULD be clear about which subset are allowed, and why.

3.4. Definition of Operations

As part of the definition of how a URI identifies a resource, a scheme definition SHOULD define the applicable set of operations that can be performed on a resource using the URI as its identifier. A model for this is HTTP methods; an HTTP resource can be operated on by GET, POST, PUT, and a number of other methods available through the HTTP protocol. The scheme definition SHOULD describe all well-defined operations on the resource identifier, and what they are supposed to do.

Some schemes don’t fit into the "information access" paradigm of URIs. For example, "telnet" provides location information for initiating a bi-directional data stream to a remote host; the only operation defined is to initiate the connection. In any case, the operations appropriate for a scheme SHOULD be documented.

Note: It is perfectly valid to say that "no operation apart from GET is defined for this URI". It is also valid to say that "there’s only one operation defined for this URI, and it’s not very GET-like". The important point is that what is defined on this scheme is described.

Scheme definitions SHOULD define a "default" operation for when a URI is invoked (or "dereferenced") by an application. For example, a common "default" operation today is to launch an application associated with the scheme name, and let it use the other URI components as inputs to do something. The default invocation, or dereferencing, of a URI SHOULD be "safe" in the sense described by section 3.4 of [W3CWebArch]; i.e., performing such an invocation should not incur any additional obligations by doing so.

3.5. Context of Use

In general, URIs are used within a broad range of protocols and applications. Most commonly, URIs are used as references to resources within directories or hypertext documents, as hyperlinks to other resources. In some cases, a scheme is intended for use within
a different, specific set of protocols or applications. If so, the
scheme definition SHOULD describe the intended use and include
references to documentation that define the applications and/or
protocols cited.

3.6. Internationalization and Character Encoding

When describing schemes in which (some of) the elements of the URI
are actually representations of human-readable text, care should be
taken not to introduce unnecessary variety in the ways in which
characters are encoded into octets and then into URI characters; see
[RFC3987] and Section 2.5 of [RFC3986] for guidelines. If URIs of a
scheme contain any text fields, the scheme definition MUST describe
the ways in which characters are encoded and any compatibility issues
with IRIs of the scheme.

The scheme specification SHOULD be as restrictive as possible
regarding what characters are allowed in the URI, because some
characters can create several different security considerations (see,
for example [RFC4690]).

All percent-encoded variants are automatically included by definition
for any character given in an IRI production. This means that if you
want to restrict the URI percent-encoded forms in some way, you must
restrict the Unicode forms that would lead to them.

3.7. Clear Security Considerations

Definitions of schemes MUST be accompanied by a clear analysis of the
security implications for systems that use the scheme; this follows
the practice of Security Consideration sections within IANA
registrations [RFC5226].

In particular, Section 7 of RFC 3986 [RFC3986] describes general
security considerations for URIs, while [RFC3987] gives those for
IRIs. The definition of an individual scheme should note which of
these apply to the specified scheme, in addition to any more scheme-
specific concerns.

3.8. Scheme Name Considerations

Section 3.1 of RFC 3986 defines the syntax of a URI scheme name; this
syntax remains the same for IRIs. New registered schemes
registrations MUST follow this syntax, which only allows a limited
repertoire of characters (taken from US-ASCII). Although the syntax
for the scheme name in URIs is case insensitive, the scheme names
itself MUST be registered using lowercase letters.
Scheme names SHOULD be short, but also sufficiently descriptive and distinguished to avoid problems.

Schemes SHOULD NOT use names or other symbols that might cause problems with rights to use the name in IETF specifications and Internet protocols. For example, be careful with trademark and service mark names. (See Section 7.4 of [RFC3978].)

Schemes SHOULD NOT use names that are either very general purpose or associated in the community with some other application or protocol. Schemes also SHOULD NOT use names that are overly general or grandiose in scope (e.g., that allude to their "universal" or "standard" nature.)

A scheme name is not a "protocol" although, like a service name as defined in section 5 of [RFC6335], it often identifies a particular protocol or application. If a scheme name has a one-to-one correspondence with a service name, then the names SHOULD be the same.

Some organizations desire their own namespace for URI scheme names for private use (see Section 6). In doing so, it is important to prevent collisions, and to make it possible to identify the owner of a private use scheme. To accomplish these two goals, such organizations SHOULD use a prefix based on their domain name, expressed in reverse order. For example, a URI scheme name of com.example.info might be used by the organization that owns the example.com domain name. Care must be taken, however, if the organization later loses the domain name embedded in their scheme names, since domain name registrations are not permanent. The URI scheme name registration procedure can be used in such an event.

Furthermore, to prevent collisions with private use scheme names, new scheme names registered MUST NOT contain a "." unless actually constructed from a reversed domain name.

4. Guidelines for Provisional URI Scheme Registration

Provisional registration can be used for schemes that are not part of any standard, but that are intended for use (or observed to be in use) that is not limited to a private environment within a single organization. Provisional registration can also be used as an intermediate step on the way to permanent registration, e.g., before the scheme specification is finalized as a standard.

For a provisional registration, the following are REQUIRED:

- The scheme name meets the syntactic requirements of Section 3.8.
o There must not already be an entry with the same scheme name. In the unfortunate case that there are multiple, different uses of the same scheme name, the Designated Expert can approve a request to modify an existing entry to note the separate use.

o Contact information identifying the person supplying the registration is included. Previously unregistered schemes discovered in use can be registered by third parties (even if not on behalf of those who created the scheme). In this case, both the registering party and the scheme creator SHOULD be identified.

o If no permanent, citable specification for the scheme definition is included, credible reasons for not providing it SHOULD be given.

o The scheme definition SHOULD include a clear Security Considerations (Section 3.7) or explain why a full security analysis is not available (e.g., in a third-party scheme registration).

o If the scheme definition does not meet the guidelines laid out in Section 3, the differences and reasons SHOULD be noted.

5. Guidelines for Historical URI Scheme Registration

In some circumstances, it is appropriate to note a scheme that was once in use or registered but for whatever reason is no longer in common use or the use is not recommended. In this case, it is possible for an individual to request that the URI scheme be registered (newly, or as an update to an existing registration) as ‘historical’. Any scheme that is no longer in common use MAY be designated as historical; the registration SHOULD contain some indication to where the scheme was previously defined or documented.

6. Guidelines for Private URI Scheme Use

Unregistered schemes can cause problems if use is not limited to a private environment within a single organization, since the use could leak out beyond the closed environment. Even within a closed environment, other colliding uses of the same scheme name could occur. As such, a unique namespace (see Section 3.8) MUST be used, and it is strongly encouraged to do a Provisional registration unless the scheme name is constructed from a domain name.
7. URI Scheme Registration Procedure

7.1. General

The IANA policy (using terms defined in [RFC5226]) for Provisional registration was formerly Expert Review and is now changed to simply use a First Come First Served policy. The policy for Permanent and Historic registration continues to be Expert Review.

The registration procedure is intended to be very lightweight for non-contentious registrations. For the most part, we expect the good sense of submitters and reviewers, guided by these procedures, to achieve an acceptable and useful consensus for the community.

In exceptional cases, where the negotiating parties cannot form a consensus, the final arbiter of any contested registration shall be the IESG.

7.2. Registration Procedures

Someone wishing to register a new scheme MUST:

1. Check the IANA URI Schemes registry to see whether there is already an entry for the desired name. If there is already an entry under the name, choose a different scheme name, or update the existing scheme specification.

2. Prepare a scheme registration request using the template specified in Section 7.4. The scheme registration request can be contained in an Internet Draft, submitted alone, or as part of some other permanently available, stable, protocol specification. The completed template can also be submitted in some other form (as part of another document or as a stand-alone document), but the completed template will be treated as an "IETF Contribution" under the guidelines of [RFC3978].

3. If the registration request is for a Permanent registration:

   1. Review the requirements in Section 3.

   2. Send a copy of the completed template or a pointer to the containing document (with specific reference to the section with the completed template) to the mailing list uri-review@ietf.org, requesting review. In addition, request review on other relevant mailing lists as appropriate. For example, general discussion of URI syntactical issues could be discussed on uri@w3.org; schemes for a network protocol
could be discussed on a mailing list for that protocol. Allow a reasonable time for discussion and comments. Four weeks is reasonable for a permanent registration request.

3. Respond to review comments and make revisions to the proposed registration as needed to bring it into line with the guidelines given in this document.

4. Submit the (possibly updated) registration template (or pointer to document containing it) to IANA at iana@iana.org.

Upon receipt of a scheme registration request, the following steps MUST be followed:

1. IANA checks the submission for completeness; if sections of the template are missing or any citations are not correct, IANA will reject the registration request.

2. If the request is for Provisional registration and no entry already exists in the current registry for the same name, IANA adds the registration to the registry, under the First Come First Served policy.

3. Otherwise, IANA enters the registration request in the IANA registry, with status marked as "Pending Review" and the remainder of this section applies.

4. IANA requests Expert Review of the registration request against the corresponding guidelines from this document.

5. The Designated Expert will evaluate the request against the criteria of the requested status.

6. In the case of a Permanent registration request, the Designated Expert may:

   * Accept the specification of the scheme for permanent registration.

   * Suggest provisional registration instead.

   * Request IETF review and IESG approval; in the meanwhile, suggest provisional registration.

   * Request additional review or discussion, as necessary.

7. If an entry already exists for the same name, the Designated Expert will determine whether the request should be rejected, or
whether the existing entry should be modified to note the separate use. This conflict process applies regardless of the requested status or the status of the existing entry.

8. Once Expert Review approves registration for a given status, IANA adds the registration to the registry.

Either based on an explicit request or independently initiated, the Designated Expert or IESG can request the upgrade of a ‘provisional’ registration to a ‘permanent’ one. In such cases, IANA will update the status of the corresponding entry. Typically this would only occur if the use is considered a standard (not necessarily an IETF standard).

7.3. Change Control

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

Provisional registrations can be updated by the original registrant or anyone designated by the original registrant. In addition, the IESG can reassign responsibility for a provisional registration scheme, or can request specific changes to a scheme registration. This will enable changes to be made to schemes where the original registrant is out of contact, or unwilling or unable to make changes.

Transition from ‘provisional’ to ‘permanent’ status can be requested and approved in the same manner as a new ‘permanent’ registration. Transition from ‘permanent’ to ‘historical’ status requires IESG approval. Transition from ‘provisional’ to ‘historical’ can be requested by anyone authorized to update the provisional registration.

7.4. URI Scheme Registration Template

This template describes the fields that MUST be supplied in a scheme registration request:

**Scheme name:**

See Section 3.8 for guidelines.

**Status:**

This reflects the status requested, and must be one of ‘permanent’, ‘provisional’, or ‘historical’.

**Applications/protocols that use this scheme name:**
See Section 3.5.

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

Author/Change controller:
Person (including contact information) authorized to change this.

References:
Include full citations for all referenced documents. Registration templates for provisional registration can be included in an Internet Draft; when the documents expire or are approved for publication as an RFC, the registration will be updated. A scheme specification is only required for Permanent registration.

The following fields are no longer required in a scheme registration request. The answers instead belong in the scheme specification.

Scheme syntax:
See Section 3.2 for guidelines.

Scheme semantics:
See Section 3.3 and Section 3.4 for guidelines.

Encoding considerations:
See Section 3.3 and Section 3.6 for guidelines.

Interoperability considerations:
If the person or group registering the scheme is aware of any details regarding the scheme that might impact interoperability, identify them here. For example: proprietary or uncommon encoding methods; inability to support multibyte character sets; incompatibility with types or versions of any underlying protocol.

Security considerations:
See Section 3.7 for guidelines.

8. The "example" Scheme

There is a need for a scheme name that can be used for examples in documentation without fear of conflicts with current or future actual schemes. The scheme "example" is hereby registered as a Permanent scheme for that purpose.

The "example" scheme is specified as follows:
Scheme syntax: The entire range of allowable syntax specified in [RFC3986] is allowed for "example" URIs.

Scheme semantics: URIs in the "example" scheme are to be used for documentation purposes only. The use of "example" URIs must not be used as locators, identify any resources, or specify any particular set of operations.

Encoding considerations: See Section 2.5 of [RFC3986] for guidelines.

Interoperability considerations: None.

Security considerations: None.

8.1. "Example" Scheme Registration Request

Scheme name: example

Status: permanent

Applications/protocols that use this scheme name: An "example" URI is to be used for documentation purposes only. It MUST NOT be used for any protocol.

Contact: N/A

Author/Change controller: IETF

References: Section 8 of this RFC XXXX.

RFC Editor Note: Replace XXXX with this RFC’s reference.

9. IANA Considerations

Previously, the former "URL Scheme" registry was replaced by the "Uniform Resource Identifier (URI) Schemes" registry. The process was based on [RFC5226] "Expert Review" with an initial (optional) mailing list review.

The updated template has an additional field for the status of the scheme, and the procedures for entering new name schemes have been augmented. Section 7 establishes the process for new scheme registration.

IANA is requested to do the following:

- Update the URI Schemes registry to point to this document.
o Combine the "Permanent URI Schemes", "Provisional URI Schemes", and "Historical URI Schemes" sub-registries into a single common registry with an additional "Status" column containing the status (Permanent, Provisional, Historical, or Pending Review), and an additional "Notes" column which is normally empty, but may contain notes approved by the Designated Expert.

o Add the "example" URI scheme to the registry (see the template in Section 8.1 for registration).

10. Security Considerations

All registered values are expected to contain accurate security consideration sections; ‘permanent’ registered scheme names are expected to contain complete definitions.

Information concerning possible security vulnerabilities of a protocol might change over time. Consequently, claims as to the security properties of a registered scheme might change as well. As new vulnerabilities are discovered, information about such vulnerabilities might need to be attached to existing documentation, so that users are not misled as to the true security properties of a registered scheme.

11. Acknowledgements

Thanks to Mark Nottingham and Graham Klyne and other members of the apps-discuss@ietf.org mailing list for their comments on this document.

Many thanks to Patrik Faltstrom, Paul Hoffmann, Ira McDonald, Roy Fielding, Stu Weibel, Tony Hammond, Charles Lindsey, Mark Baker, and other members of the uri@w3.org mailing list for their comments on earlier versions.

Parts of this document are based on [RFC2717], [RFC2718] and [RFC3864]. Some of the ideas about use of URIs were taken from the "Architecture of the World Wide Web" [W3CWebArch].

12. References

12.1. Normative References


12.2. Informative References


Appendix A. Changes Since RFC 4395

1. Combined the Historical, Permanent, and Provisional URI Schemes registries into one registry with a status column. This is done to make it easier to prevent duplicates and see existing conventions.

2. Added a Notes column in the registry for notes approved by the Designated Expert.

3. Moved the following fields out of the scheme registration request template and into the requirements for a scheme specification: Scheme syntax, Scheme semantics, Encoding considerations, Interoperability considerations, and Security considerations.

4. Simplified the process for Provisional registration significantly: changed from Expert Review to First Come First Served, and clarified that mailing list review is not required.

5. Updated process for handling of scheme name conflicts, so that adding a note can be approved by the Designated Expert rather than the IESG.

6. Clarified that a "URI scheme name" and an "IRI scheme name" are the same thing and thus use the same IANA registry.

7. Clarified that a registration request falls under the "IETF Contribution" rules, but the scheme’s specification need not.

8. Added the "example:" URI scheme.

9. Added text about when to use Provisional registration.

10. Updated convention for Private use schemes to use "." (instead of "-" ) between domain name labels, to reduce chance of collision, and recommended use of a reverse domain name prefix to allow identifying the owning organization.

11. Recommended that scheme definitions define a "default" operation for when a URI is invoked.

12. Recommended that a scheme name be the same as the service name, when there exists a 1:1 correspondence.

13. Elaborated on when a Provisional request should be upgraded to Permanent.
Authors’ Addresses

Dave Thaler (editor)
Microsoft
One Microsoft Way
Redmond, WA  98052
US
Phone: +1 425 703 8835
Email: dthaler@microsoft.com

Tony Hansen
AT&T Laboratories
200 Laurel Ave.
Middletown, NJ  07748
USA
Email: tony+urireg@mailennium.att.com

Ted Hardie
Google
Phone: +1 408 628 5864
Email: ted.ietf@gmail.com

Larry Masinter
Adobe
345 Park Ave.
San Jose, CA  95110
US
Phone: +1 408 536 3024
Email: masinter@adobe.com
URI:  http://larry.masinter.net