

Network Working Group
Internet-Draft
Obsoletes: 4288 (if approved)
Expires: March 8, 2012

N. Freed
Oracle
J. Klensin

T. Hansen
AT&T Laboratories
September 5, 2011

Media Type Specifications and Registration Procedures
draft-freed-media-type-regs-01

Abstract

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

Historical Note

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

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

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 8, 2012.

Copyright Notice

Copyright (c) 2011 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	5
2. Media Type Registration Preliminaries	5
3. Registration Trees and Subtype Names	5
3.1. Standards Tree	6
3.2. Vendor Tree	6
3.3. Personal or Vanity Tree	7
3.4. Special x. Tree	7
3.5. Additional Registration Trees	8
4. Registration Requirements	8
4.1. Functionality Requirement	8
4.2. Naming Requirements	8
4.2.1. Text Media Types	9
4.2.2. Image Media Types	10
4.2.3. Audio Media Types	10
4.2.4. Video Media Types	10
4.2.5. Application Media Types	11
4.2.6. Multipart and Message Media Types	11
4.2.7. Additional Top-level Types	11
4.2.8. Structured Syntax Name Suffixes	12
4.3. Parameter Requirements	12
4.4. Canonicalization and Format Requirements	13
4.5. Interchange Recommendations	14
4.6. Security Requirements	14
4.7. Requirements specific to XML media types	15
4.8. Encoding Requirements	15
4.9. Usage and Implementation Non-requirements	16
4.10. Publication Requirements	17
4.11. Additional Information	17
5. Media Type Registration Procedures	18
5.1. Preliminary Community Review	18
5.2. Review and Approval	19
5.3. IANA Registration	19
5.4. Media Types Reviewer	19
5.5. Comments on Media Type Registrations	20
5.6. Location of Registered Media Type List	20
5.7. IANA Procedures for Registering Media Types	20
5.8. Change Procedures	21
5.9. Registration Template	22
6. Structured Syntax Suffix Registration Procedures	23
6.1. Change Procedures	24
6.2. Structured Syntax Suffix Registration Template	24
7. Security Considerations	25
8. IANA Considerations	25
9. Acknowledgements	25
10. References	25
10.1. Normative References	25

10.2. Informative References	26
Appendix A. Grandfathered Media Types	26
Appendix B. Changes Since RFC 4288	27
Authors' Addresses	27

1. Introduction

Recent Internet protocols have been carefully designed to be easily extensible in certain areas. In particular, many protocols, including but not limited to MIME [RFC2045], are capable of carrying arbitrary labeled content. A mechanism is needed to label such content and a registration process is needed for these labels, to ensure that the set of such values is developed in an orderly, well-specified, and public manner.

This document defines media type specification and registration procedures that use the Internet Assigned Numbers Authority (IANA) as a central registry.

1.1. 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].

This specification makes use of the Augmented Backus-Naur Form (ABNF) [RFC4234] notation, including the core rules defined in Appendix A of that document.

2. Media Type Registration Preliminaries

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

3. Registration Trees and Subtype Names

In order to increase the efficiency and flexibility of the registration process, different structures of subtype names may be registered to accommodate the different natural requirements for, e.g., a subtype that will be recommended for wide support and implementation by the Internet community, or a subtype that is used to move files associated with proprietary software. The following subsections define registration "trees" that are distinguished by the use of faceted names, e.g., names of the form "tree.subtree...subtype". Note that some media types defined prior

to this document do not conform to the naming conventions described below. See Appendix A for a discussion of them.

3.1. Standards Tree

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

1. approved directly by the IESG, or
2. registered using the "Specification Required" IANA registration policy [RFC5226] (which implies Expert Review).

The former procedure is used for registering registrations from IETF Consensus documents, or in rare cases when registering a grandfathered (see Appendix A) and/or otherwise incomplete registration is in the interest of the Internet community. In the latter case the IESG makes the decision on whether the registration submitter represents a recognized Standards Body; after that a Media Types Reviewer (Designated Expert or a group of Designated Experts) performs the expert review as specified in this document.

In the case of registration for the IETF itself, the registration proposal **MUST** be published as an IETF Consensus RFC.

Registrations published in non-IETF RFC streams are allowed and require IESG approval.

Standards-tree registration RFCs can either be standalone "registration only" RFCs, or they can be incorporated into a more general specification of some sort.

Media types in the standards tree are normally denoted by names that are not explicitly faceted, i.e., do not contain period (".", full stop) characters.

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

3.2. Vendor Tree

The vendor tree is used for media types associated with commercially available products. "Vendor" or "producer" are construed as

equivalent and very broadly in this context.

A registration may be placed in the vendor tree by anyone who needs to interchange files associated with the particular product. However, the registration formally belongs to the vendor or organization producing the software or file format being registered. Changes to the specification will be made at their request, as discussed in subsequent sections.

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

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

3.3. Personal or Vanity Tree

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

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

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

3.4. Special x. Tree

For convenience and symmetry with this registration scheme, subtype names with "x." as the first facet may be used for the same purposes for which names starting in "x-" are used. These types are unregistered, experimental, and for use only with the active agreement of the parties exchanging them.

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

Types in this tree MUST NOT be registered. If a generally useful and widely deployed type incorrectly ends up with an "x." or "x-" name prefix, it MAY be registered using its current name in an alternate tree by following the procedure defined in Appendix A.

3.5. Additional Registration Trees

From time to time and as required by the community, the IANA may, by and with the advice and consent of the IESG, create new top-level registration trees. It is explicitly assumed that these trees may be created for external registration and management by well-known permanent bodies; for example, scientific societies may register media types specific to the sciences they cover. In general, the quality of review of specifications for one of these additional registration trees is expected to be equivalent to registrations in the standards tree. Establishment of these new trees MUST be done with a Standards Track RFC.

4. Registration Requirements

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

4.1. Functionality Requirement

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

This requirement applies regardless of the registration tree involved.

4.2. Naming Requirements

All registered media types MUST be assigned type and subtype names. The combination of these names serves to uniquely identify the media type, and the format of the subtype name identifies the registration

tree. Both type and subtype names are case-insensitive.

Type and subtype names beginning with "X-" are reserved for experimental use and MUST NOT be registered. This parallels the restriction on the x. tree, as discussed in Section 3.4.

Type and subtype names MUST conform to the following ABNF:

```
type-name = reg-name
subtype-name = reg-name

reg-name = 1*127reg-name-chars
reg-name-chars = ALPHA / DIGIT / "!" /
                "#" / "$" / "&" / "." /
                "+" / "-" / "^" / "_"
```

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

Although the name syntax treats "+" as equivalent to any other character, it is used in media type names to introduce a structured syntax specifier suffix. Structured syntax suffix requirements are specified in Section 4.2.8.

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

These requirements apply regardless of the registration tree involved.

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

4.2.1. Text Media Types

The "text" media type is intended for sending material that is principally textual in form. A "charset" parameter MAY be used to indicate the charset of the body text for "text" subtypes, notably including the subtype "text/plain", which is a generic subtype for plain text defined in [RFC2046]. If defined, a text "charset" parameter MUST be used to specify a charset name defined in

accordance to the procedures laid out in [RFC2978].

Plain text does not provide for or allow formatting commands, font attribute specifications, processing instructions, interpretation directives, or content markup. Plain text is seen simply as a linear sequence of characters, possibly interrupted by line breaks or page breaks. Plain text MAY allow the stacking of several characters in the same position in the text. Plain text in scripts like Arabic and Hebrew may also include facilities that allow the arbitrary mixing of text segments with opposite writing directions.

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

4.2.2. Image Media Types

A media type of "image" indicates that the content specifies one or more separate images that require appropriate hardware to display. The subtype names the specific image format.

4.2.3. Audio Media Types

A media type of "audio" indicates that the content contains audio data.

4.2.4. Video Media Types

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

Note that although in general this document strongly discourages the mixing of multiple media in a single body, it is recognized that many so-called video formats include a representation for synchronized audio and/or text, and this is explicitly permitted for subtypes of "video".

4.2.5. Application Media Types

The "application" media type is to be used for discrete data that do not fit in any of the media types, and particularly for data to be processed by some type of application program. This is information that must be processed by an application before it is viewable or usable by a user. Expected uses for the "application" media type include but are not limited to file transfer, spreadsheets, presentations, scheduling data, and languages for "active" (computational) material. (The latter, in particular, can pose security problems that must be understood by implementors, and are considered in detail in the discussion of the "application/PostScript" media type in [RFC2046].)

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

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

4.2.6. Multipart and Message Media Types

Multipart and message are composite types, that is, they provide a means of encapsulating zero or more objects, each labeled with its own media type.

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

4.2.7. Additional Top-level Types

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

4.2.8. Structured Syntax Name Suffixes

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

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

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

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

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

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

4.3. Parameter Requirements

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

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

parameter-name = reg-name

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

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

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

4.4. Canonicalization and Format Requirements

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

A precise and openly available specification of the format of each media type **MUST** exist for all types registered in the standards tree and **MUST** at a minimum be referenced by, if it isn't actually included in, the media type registration proposal itself.

The specifications of format and processing particulars may or may not be publicly available for media types registered in the vendor tree, and such registration proposals are explicitly permitted to limit specification to which software and version produce or process such media types. References to or inclusion of format specifications in registration proposals is encouraged but not required.

Format specifications are still required for registration in the personal tree, but may be either published as RFCs or otherwise deposited with the IANA. The deposited specifications will meet the same criteria as those required to register a well-known TCP port and, in particular, need not be made public.

Some media types involve the use of patented technology. The registration of media types involving patented technology is specifically permitted. However, the restrictions set forth in [RFC2026] on the use of patented technology in IETF standards-track protocols must be respected when the specification of a media type is

part of a standards-track protocol. In addition, other standards bodies making use of the standards tree may have their own rules regarding intellectual property that must be observed in their registrations.

4.5. Interchange Recommendations

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

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

These recommendations apply regardless of the registration tree involved.

4.6. Security Requirements

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

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

The security considerations section of all registrations is subject to continuing evaluation and modification, and in particular MAY be extended by use of the "comments on media types" mechanism described in Section 5.5 below.

Some of the issues that should be looked at in a security analysis of a media type are:

- o Complex media types may include provisions for directives that institute actions on a recipient's files or other resources. In many cases provision is made for originators to specify arbitrary actions in an unrestricted fashion that may then have devastating effects. See the registration of the application/postscript media type in [RFC2046] for an example of such directives and how they should be described in a media type registration.
- o All registrations MUST state whether or not they employ such "active content", and if they do, they MUST state what steps have been taken to protect users of the media type from harm.
- o Complex media types may include provisions for directives that institute actions that, while not directly harmful to the recipient, may result in disclosure of information that either facilitates a subsequent attack or else violates a recipient's privacy in some way. Again, the registration of the application/postscript media type illustrates how such directives can be handled.
- o A media type that employs compression may provide an opportunity for sending a small amount of data that, when received and evaluated, expands enormously to consume all of the recipient's resources. All media types SHOULD state whether or not they employ compression, and if they do they should discuss what steps need to be taken to avoid such attacks.
- o A media type might be targeted for applications that require some sort of security assurance but not provide the necessary security mechanisms themselves. For example, a media type could be defined for storage of confidential medical information that in turn requires an external confidentiality service, or which is designed for use only within a secure environment.

4.7. Requirements specific to XML media types

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

4.8. Encoding Requirements

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

It is therefore useful to note what sort of data a media type can

consist of as part of its registration. An "encoding considerations" field is provided for this purpose. Possible values of this field are:

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

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

binary: The content consists of unrestricted sequence of octets.

framed: The content consists of a series of frames or packets without internal framing or alignment indicators. Additional out-of-band information is needed to interpret the data properly, including but not necessarily limited to, knowledge of the boundaries between successive frames and knowledge of the transport mechanism. Note that media types of this sort cannot simply be stored in a file or transported as a simple stream of octets; therefore, such media types are unsuitable for use in many traditional protocols. A commonly used transport with framed encoding is the Real-time Transport Protocol, RTP. Additional rules for framed encodings defined for transport using RTP are given in [RFC3555].

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

4.9. Usage and Implementation Non-requirements

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

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

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

purpose.

4.10. Publication Requirements

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

As stated previously, standards tree registrations for media types defined in documents produced by other standards bodies MUST be described by a formal standards specification produced by that body. Such specifications MUST contain an appropriate media type registration template taken from Section 5.9. Additionally, the copyright on the registration template MUST allow the IANA to copy it into the IANA registry.

Other than IETF registrations in the standards tree, the registration of a data type does not imply endorsement, approval, or recommendation by the IANA or the IETF or even certification that the specification is adequate. To become Internet Standards, a protocol or data object must go through the IETF standards process. This is too difficult and too lengthy a process for the convenient registration of media types.

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

As discussed above, registration of a top-level type requires standards-track processing in the IETF and, hence, RFC publication.

4.11. Additional Information

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

- o Magic number(s) (length, octet values). Magic numbers are byte sequences that are always present at a given place in the file and thus can be used to identify entities as being of a given media type.

- o File name extension(s) commonly used on one or more platforms to indicate that some file contains a given media type.
- o Mac OS File Type code(s) (4 octets) used to label files containing a given media type.
- o Information about how fragment/anchor identifiers [RFC3986] are constructed for use in conjunction with this media type.

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

5. Media Type Registration Procedures

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

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

Registrations in the vendor and personal tree should be submitted directly to the IANA, ideally after first posting to the `ietf-types@iana.org` list for review.

Proposed registrations in the standards tree by other standards bodies MUST be communicated to both the IESG (at `iesg@ietf.org`) and to the `ietf-types` list (at `ietf-types@iana.org`). Prior posting as an Internet Draft is not required for these registrations, but may be helpful to the IESG and is encouraged.

5.1. Preliminary Community Review

Notice of a potential media type registration in the standards tree MUST be sent to the "`ietf-types@iana.org`" mailing list for review. This mailing list has been established for the purpose of reviewing proposed media and access types. Registrations in other trees MAY be sent to the list for review as well.

The intent of the public posting to this list is to solicit comments and feedback on the choice of type/subtype name, the unambiguity of the references with respect to versions and external profiling information, and a review of any interoperability or security

considerations. The submitter may submit a revised registration or abandon the registration completely and at any time.

5.2. Review and Approval

Media types registered in the standards tree MUST either be reviewed and approved by the IESG or follow the Specification Required process:

1. Specification in a permanent and readily available public specification
2. Review and approval by the Designated Expert for media types

5.3. IANA Registration

Provided that the media type meets all of the relevant requirements and has obtained whatever approval this specification requires, the author may submit the registration request to the IANA. Registration requests can be sent to iana@iana.org. A web form for registration requests is also available:

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

Sending to ietf-types@iana.org does not constitute submitting the registration to the IANA.

When the registration is either part of an RFC publication request or a registration in the standards tree submitted to the IESG, close coordination between the IANA and the IESG means IESG approval in effect submits the registration to the IANA. There is no need for an additional registration request in such cases.

5.4. Media Types Reviewer

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

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

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

5.5. Comments on Media Type Registrations

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

5.6. Location of Registered Media Type List

Media type registrations are listed by the IANA at:

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

5.7. IANA Procedures for Registering Media Types

Upon receipt of a registration request, the IANA will submit the request for Expert Review. The Expert Reviewer will check to see that the following minimal conditions are met:

- o Media types MUST function as an actual media format. In particular, charsets and transfer encodings MUST NOT be registered as media types.
- o All media types MUST have properly formed type and subtype names. All type names MUST be defined by a standards-track RFC. All type/subtype name pairs MUST be unique and MUST contain the proper tree prefix.
- o Types registered in the standards and personal trees MUST either provide a format specification or a pointer to one.
- o All media types MUST have a reasonable security considerations section. (It is neither possible nor necessary for the IANA to conduct a comprehensive security review of media type registrations. Nevertheless, the IANA has the authority to identify obviously incompetent material and return it to the submitter for revision.)
- o Registrations in the standards tree MUST satisfy the additional requirement that they originate from the IETF itself or from another Standards Body recognized as such by the IETF.

Registrations in the standards tree MUST satisfy the additional requirement that they originate from the IETF itself or from another Standards Body recognized as such by the IETF. The IESG determines

whether or not a given organization qualifies as a standards body.

5.8. Change Procedures

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

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

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

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

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

5.9. Registration Template

To: ietf-types@iana.org

Subject: Registration of media type XXX/YYY

Type name:

Subtype name:

Required parameters:

Optional parameters:

Encoding considerations:

Security considerations:

Interoperability considerations:

Published specification:

Applications that use this media type:

Additional information:

 Magic number(s):

 File extension(s):

 Macintosh file type code(s):

 URI fragment/anchor identifier(s):

Person & email address to contact for further information:

Intended usage:

(One of COMMON, LIMITED USE or OBSOLETE.)

Restrictions on usage:

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

Author:

Change controller:

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

Some discussion of Macintosh file type codes and their purpose can be

found in [MacOSFileTypes]. Additionally, please refrain from writing "none" or anything similar when no file extension or Macintosh file type is specified, lest "none" be confused with an actual code value.

6. Structured Syntax Suffix Registration Procedures

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

1. Check IANA's registry of media type name suffixes to see whether or not there is already an entry for that well-defined structured syntax.
2. If there is no entry for their suffix scheme, fill out the template (specified in Section 6.2) and include that with the media type registration. The template may be contained in an Internet Draft, alone or as part of some other protocol specification. The template may also be submitted in some other form (as part of another document or as a stand-alone document), but the contents will be treated as an "IETF Contribution" under the guidelines of RFC 3978 [RFC3978].
3. Send a copy of the template or a pointer to the containing document (with specific reference to the section with the template) to the mailing list ietf-types@ietf.org, requesting review. This may be combined with a request to review the media type registration. Allow a reasonable time for discussion and comments.
4. Respond to review comments and make revisions to the proposed registration as needed to bring it into line with the guidelines given in this document.
5. Submit the (possibly updated) registration template (or pointer to document containing it) to IANA at iana@iana.org.

Upon receipt of a structured syntax suffix registration request,

1. IANA checks the submission for completeness; if sections are missing or citations are not correct, IANA rejects the registration request.
2. IANA checks the current registry for a entry with the same name; if such a registry exists, IANA rejects the registration request.
3. IANA requests Expert Review of the registration request against the corresponding guidelines.

4. The Designated Expert may request additional review or discussion, as necessary.
5. If Expert Review recommends registration, IANA adds the registration to the appropriate registry.

6.1. Change Procedures

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

6.2. Structured Syntax Suffix Registration Template

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

Name

Full name of the well-defined structured syntax.

+suffix

Suffix used to indicate conformance to the syntax.

References.

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

Encoding considerations

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

Interoperability considerations

Any issues regarding the interoperable use of types employing this structured syntax should be given here. Examples would include the existence of incompatible versions of the syntax, issues combining certain charsets with the syntax, or incompatibilities with other types or protocols.

Security considerations

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

Contact

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

Author/Change controller.

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

7. Security Considerations

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

8. IANA Considerations

The purpose of this document is to define IANA registries for media types and structured syntax suffixes.

9. Acknowledgements

The current authors would like to acknowledge their debt to the late Dr. Jon Postel, whose general model of IANA registration procedures and specific contributions shaped the predecessors of this document [RFC2048]. We hope that the current version is one with which he would have agreed but, as it is impossible to verify that agreement, we have regretfully removed his name as a co-author.

Alexey Melnikov provided many helpful review comments and suggestions.

10. References

10.1. Normative References

- [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.
- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

- [RFC2978] Freed, N. and J. Postel, "IANA Charset Registration Procedures", BCP 19, RFC 2978, October 2000.
- [RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.
- [RFC3555] Casner, S. and P. Hoschka, "MIME Type Registration of RTP Payload Formats", RFC 3555, July 2003.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

10.2. Informative References

- [MacOSFileTypes]
Apple Computer, Inc., "Mac OS: File Type and Creator Codes, and File Formats", Apple Knowledge Base Article 55381, June 1993,
<<http://www.info.apple.com/kbnum/n55381>>.
- [RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.
- [RFC2048] Freed, N., Klensin, J., and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 2048, November 1996.
- [RFC2231] Freed, N. and K. Moore, "MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations", RFC 2231, November 1997.
- [RFC3978] Bradner, S., "IETF Rights in Contributions", RFC 3978, March 2005.

Appendix A. Grandfathered Media Types

A number of media types with unfaceted names, registered prior to 1996, would, if registered under the guidelines in this document, be

given a faceted name and placed into either the vendor or personal trees. Reregistration of those types to reflect the appropriate trees is encouraged but not required. Ownership and change control principles outlined in this document apply to those types as if they had been registered in the trees described above.

From time to time there may also be cases where a media type with an unfaceted name has been widely deployed without being registered. If possible such types SHOULD be reregistered with a proper faceted name. However, if this is not possible the type can, subject to approval by both the media types reviewer and the IESG, be registered in the proper tree with its unfaceted name.

Appendix B. Changes Since RFC 4288

- o Suffixes to indicate the use of a particular structured syntax are now fully specified and a suffix registration process has been defined.
- o Registration of widely deployed unregistered unfaceted type names in the vendor or personal trees is now allowed, subject to approval by the media types reviewer and the IESG.
- o The standards tree registration process has been revised to include Expert Review and generalized to address cases like media types in non-IETF stream documents.
- o A field for fragment/anchor identifiers has been added to the registration template.

Authors' Addresses

Ned Freed
Oracle
800 Royal Oaks
Monrovia, CA 91016-6347
USA

Email: ned+ietf@mrochek.com

John C. Klensin
1770 Massachusetts Ave, #322
Cambridge, MA 02140
USA

Email: klensin+ietf@jck.com

Tony Hansen
AT&T Laboratories
200 Laurel Ave.
Middletown, NJ 07748
USA

Email: tony+mtsuffix@maillennium.att.com

Individual submission
Internet-Draft
Intended status: BCP
Expires: January 12, 2012

M. Kucherawy
Cloudmark, Inc.
July 11, 2011

Best Current Practices for Handling of Malformed Messages
draft-kucherawy-mta-malformed-03

Abstract

The email ecosystem has long had a very permissive set of common processing rules in place, despite increasingly rigid standards governing its components, ostensibly to improve the user experience. The handling of these come at some cost, and various components are faced with decisions about whether or not to permit non-conforming messages to continue toward their destinations unaltered, adjust them to conform (possibly at the cost of losing some of the original message), or outright rejecting them.

This memo includes a collection of the best current practices in a variety of such situations, to be used as implementation guidance. It must be emphasized, however, that the intent of this memo is not to standardize malformations or otherwise encourage their proliferation. The messages that are the subject of this memo are manifestly malformed, and the code and culture that generates them needs to be fixed. Nevertheless, many malformed messages from otherwise legitimate senders are in circulation and will be for some time and, unfortunately, commercial reality shows that we cannot simply reject or discard them. Accordingly, this memo presents recommendations for dealing with them in ways that seem to do the least additional harm until the infrastructure is tightened up to match the standards.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 12, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. The Purpose Of This Work	3
1.2. Not The Purpose Of This Work	3
2. Keywords	3
3. Background	4
4. Internal Representations	4
5. Mail Submission Agents	4
6. Header Anomalies	4
6.1. Non-Header Lines	5
6.2. Header Malformations	6
6.3. Header Field Counts	6
7. MIME Anomalies	7
7.1. Missing MIME-Version Field	7
8. IANA Considerations	8
9. Security Considerations	8
10. References	8
10.1. Normative References	8
10.2. Informative References	8
Appendix A. Examples	9
Appendix B. Acknowledgements	9

1. Introduction

1.1. The Purpose Of This Work

The history of email standards, going back to [RFC822] and beyond, contains a fairly rigid evolution of specifications. But implementations within that culture have also long had an undercurrent known formally as the robustness principle, but also known informally as Postel's Law: "Be conservative in what you do, be liberal in what you accept from others."

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

This memo presents some areas in which the more lenient stances can provide vectors for attack, and then presents the collected wisdom of numerous applications in and around the email ecosystem for dealing with them to mitigate their impact.

1.2. Not The Purpose Of This Work

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

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

2. Keywords

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

3. Background

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

4. Internal Representations

Any agent handling a message could have one or two (or more) distinct representations of a message it is handling. One is an internal representation, such as a block of storage used for the header and a block for the body. These may be sorted, encoded, decoded, etc. as per the needs of that particular module. The other is the representation that is output to the next agent in the handling chain. This might be identical to the version that is input to the module, or it might have some changes such as added or reordered header fields, body modifications to remove malicious content, etc.

In some cases, advice is provided only for internal representations. However, there is often occasion to mandate changes to the output as well.

5. Mail Submission Agents

Within the email context, the single most influential component that can reduce the presence of malformed items in the email system is the Mail Submission Agent (MSA). This is the component that is essentially the interface between end users that create content and the mail stream.

The lax processing described earlier in the document creates a high support and security cost overall. Thus, MSAs MUST evolve to become more strict about enforcement of all relevant email standards, especially [MAIL] and the [MIME] family of documents.

Relay Mail Transport Agents (MTAs) SHOULD also be more strict; although preventing the dissemination of malformed messages is desirable, the rejection of such mail already in transit also has a support cost, namely the creation of a [DSN] that many end users might not understand.

6. Header Anomalies

This section covers common syntactical and semantic anomalies found in headers of messages, and presents preferred mitigations.

6.1. Non-Header Lines

It has been observed that some messages contain a line of text in the header that is not a valid message header field of any kind. For example:

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

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

The cause of this is typically a bug in a message generator of some kind. If the fourth line was intended to be a continuation of the third, it should be indented by whitespace as set out in Section 2.2.3 of [MAIL].

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

1. some agents choose to separate the header of the message from the body only at the first empty line (i.e. a CRLF immediately followed by another CRLF);
2. some agents assume this anomaly should be interpreted to mean the body starts at line four, as the end of the header is assumed by encountering something that is not a valid header field or folded portion thereof;
3. some agents assume this should be interpreted as an intended header folding as described above;
4. some agents reject this outright as line four is neither a valid header field nor a folded continuation of a header field prior to an empty line.

This can be exploited if it is known that one message handling agent will take one action while the next agent in the handling chain will take another. For example, a filter trained to detect malicious body anomalies (e.g. references to dangerous web sites) that is fed by a Mail Transfer Agent (MTA) implementing (1) above might not get the opportunity to identify something dangerous in a message if it is unaware of the anomaly and does not itself check for it.

Consensus indicates the preferred implementation is to terminate header processing before the first character in line four, as

described in (2) above. Thus, a module compliant with this specification MUST terminate header processing upon encountering the first line of text that is not a valid header field. That is, all data after that point in the input MUST NOT be considered part of the header of the message. If that line is not an empty line, an empty line MUST be inserted at that point in the emitted version of the message being processed.

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

6.2. Header Malformations

There are various malformations that exist. A common one is insertion of whitespace at unusual locations, such as:

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

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

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

The acceptance grammar of [MAIL] permits that extra whitespace, so it cannot be considered invalid. However, a consensus of implementations prefers to remove that whitespace. There is no perceived change to the semantics of the header field being altered as the whitespace is itself semantically meaningless. Thus, a module compliant with this memo MUST remove all whitespace after the field name but before the colon, and MUST emit that version of that field on output.

6.3. Header Field Counts

Section 3.6 of [MAIL] prescribes specific header field counts for a valid message. Few agents actually enforce these in the sense that a message whose header contents exceed one or more limits set there are generally allowed to pass; they may add any required fields that are missing, however.

Also, few agents that use messages as input, including Mail User

Agents (MUAs) that actually display messages to users, verify that the input is valid before proceeding. Two popular open source filtering programs and two popular Mailing List Management (MLM) packages examined at the time this memo was drafted select either the first or last instance of a particular field name, such as From, to decide who sent a message. Absent enforcement of [MAIL], an attacker can craft a message with multiple fields if that attacker knows the filter will make a decision based on one but the user will be shown the other.

This situation is exacerbated when a claim of message validity is inferred by something like a valid [DKIM] signature. Such a signature might cover one instance of a constrained field but not another, and a naive consumer of DKIM's output, not realizing which one was covered by a valid signature, presume the wrong one was the "good" one. An MUA, for example could show the first of two From fields as "good" or "safe" while the DKIM signature actually only verified the second.

Thus, an agent compliant with this specification MUST enact one of the following:

1. reject outright or refuse to process further any input message that does not conform to Section 3.6 of [MAIL];
2. remove or, in the case of an MUA, refuse to render any instances of a header field whose presence exceeds a limit prescribed in Section 3.6 of [MAIL] when generating its output;
3. alter the name of any header field whose presence exceeds a limit prescribed in Section 3.6 of [MAIL] when generating its output so that later agents can produce a consistent result.

7. MIME Anomalies

[MIME], et seq, define a mechanism of message extensions for providing text in character sets other than ASCII, non-text attachments to messages, multi-part message bodies and similar facilities.

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

7.1. Missing MIME-Version Field

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

It is often observed that a message has complete MIME structure, yet lacks this header field.

As described at the end of Section 6.1, this is not expected from a reputable content generator and is often an indication of mass-produced spam or other undesirable messages.

Therefore, an agent compliant with this specification MUST internally enact one or more of the following in the absence of a MIME-Version header field:

1. Ignore all other MIME-specific fields, even if they are syntactically valid, thus treating the entire message as a single-part message of type text/plain;
2. Remove all other MIME-specific fields, even if they are syntactically valid, both internally and when emitting the output version of the message;
3. Rename all other MIME-specific fields, even if they are syntactically valid, both internally and when emitting the output version of the message.

8. IANA Considerations

This memo contains no actions for IANA.

9. Security Considerations

The discussions of the anomalies above and their prescribed solutions are themselves security considerations. The practises enumerated in this memo are generally perceived to resolve security considerations that already exist rather than introducing new ones.

10. References

10.1. Normative References

- [KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [MAIL] Resnick, P., "Internet Message Format", RFC 5322, October 2008.

10.2. Informative References

- [DKIM] Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., and M. Thomas, "DomainKeys Identified Mail (DKIM)

Signatures", RFC 4871, May 2007.

- [DSN] Moore, K. and G. Vaudreuil, "An Extensible Message Format for Delivery Status Notifications", RFC 3464, January 2003.
- [EMAIL-ARCH] Crocker, D., "Internet Mail Architecture", RFC 5598, July 2009.
- [MIME] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.
- [RFC822] Crocker, D., "Standard for the Format of Internet Text Messages", RFC 822, August 1982.

Appendix A. Examples

Examples, if needed, can go here.

Appendix B. Acknowledgements

The author wishes to acknowledge the following for their review and constructive criticism of this proposal: (names)

Author's Address

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

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

Individual Submission
Internet-Draft
Updates: 1123 (if approved)
Intended status: Standards Track
Expires: June 1, 2012

L-J. Liman
Netnod
J. Abley
ICANN
November 29, 2011

Top Level Domain Name Specification
draft-liman-tld-names-06

Abstract

The syntax for allowed Top-Level Domain (TLD) labels in the Domain Name System (DNS) is not clearly applicable to the encoding of Internationalised Domain Names (IDNs) as TLDs.

This document provides a concise specification of TLD label syntax based on existing syntax documentation, extended minimally to accommodate IDNs.

This document updates RFC1123.

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 June 1, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	3
2. Definitions	4
3. Background	5
4. TLD Label Syntax Specification	7
5. Policy Considerations	8
6. IANA Considerations	9
7. Security Considerations	10
8. Acknowledgements	11
9. References	12
9.1. Normative References	12
9.2. Informative References	12
Appendix A. Change History	13
A.1. draft-liman-tld-names-06	13
A.2. draft-liman-tld-names-05	13
A.3. draft-liman-tld-names-04	13
A.4. draft-liman-tld-names-03	13
A.5. draft-liman-tld-names-02	13
A.6. draft-liman-tld-names-01	13
A.7. draft-liman-tld-names-00	13
A.8. Version Identification Tag	14
Authors' Addresses	15

1. Introduction

The syntax of TLD labels ("TLD DNS-Labels", as defined in Section 2) is specified in [RFC1123], where such labels are asserted to be "alphabetic" within a section of that document entitled "DISCUSSION". This can be interpreted as requiring that the hyphen character ("-") and numeric digits be excluded from TLD DNS-Labels. Such a restriction would not accommodate the US-ASCII encoding of Internationalised Domain Names (IDNs), as specified in [RFC5890]. A more detailed discussion of the existing specifications can be found in Section 3.

This document extends the syntax of allowable TLD DNS-Labels to support IDNs, but places some restrictions on the choice of IDN labels. These restrictions are intended to be consistent with the existing specification for US-ASCII TLD DNS-Labels. See Section 4 for the updated specification.

This document focuses narrowly on the issue of allowable DNS-Labels in TLDs and does not (and is not intended to) make any other changes or clarifications to existing domain name syntax rules.

It is carefully noted that the specification in this document is not the only factor in choosing suitable TLD DNS-Labels, and that many considerations external to the IETF are included in that wider policy. See Section 5 for more discussion of policy considerations.

2. Definitions

The term DNS-Label is used in this document to have precisely the same meaning as the term "label", as introduced in [RFC1034], section 3.1. A DNS-Label denotes one node in a DNS tree. A DNS-Label is zero to 63 octets in length. The term "DNS-Label" refers exclusively to the "wire format" of the label, and not to any presentation format of the label.

A Top-Level Domain (TLD) DNS-Label is the right-most ("highest-level") DNS-Label in a fully-qualified domain name.

The terms A-Label and U-Label are used in this document as defined in [RFC5890].

3. Background

[RFC0952] defines a host name as follows:

'A "name" ... is a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9), minus sign (-), and period (.). Note that periods are only allowed when they serve to delimit components of "domain style names". (See RFC-921, "Domain Name System Implementation Schedule", for background). No blank or space characters are permitted as part of a name. No distinction is made between upper and lower case. The first character must be an alpha character. The last character must not be a minus sign or period.' [Unnumbered section titled "ASSUMPTIONS", first paragraph]

[RFC1123] reaffirms this definition, but makes one change to the syntax:

'The syntax of a legal Internet host name was specified in RFC-952 [DNS:4]. One aspect of host name syntax is hereby changed: the restriction on the first character is relaxed to allow either a letter or a digit. Host software MUST support this more liberal syntax.' [Section 2.1]

In addition, the DISCUSSION section of Section 2.1 says:

'However, a valid host name can never have the dotted-decimal form #.#.#.#, since at least the highest-level component label will be alphabetic.' [Section 2.1]

Some implementers may have understood the above phrase 'will be alphabetic' to be a protocol restriction.

Neither [RFC0952] nor [RFC1123] explicitly states the reasons for these restrictions. It might be supposed that human factors were a consideration; [RFC1123] appears to suggest that one of the reasons was to prevent confusion between dotted-decimal IPv4 addresses and host domain names. In any case, it is reasonable to believe that the restrictions have been assumed in some deployed software, and that changes to the rules should be undertaken with caution.

The Internationalised Domain Names in Applications 2008 specification (IDNA2008) [RFC5891] [RFC5892] provides a protocol for encoding Unicode strings in DNS-Labels. The Unicode string used by applications is known as a U-Label; its corresponding encoding in the DNS is known as an A-Label. The terms A-Label and U-Label are used in this document as defined in [RFC5890]. Valid A-Labels always contain non-alphabetic characters.

In order to accommodate the wish to express TLD names in scripts other than Latin (or rather, the US-ASCII subset of Latin), it is necessary to allow non-alphabetic characters in the corresponding TLD DNS-Labels. To minimize changes, the U-label form of a TLD name is restricted in ways functionally compatible with the restrictions (from [RFC0952] and [RFC1123]) on US-ASCII TLD names, by applying rules analogous to those already imposed on US-ASCII TLD DNS-Labels to TLD U-labels.

However, deployed software that checks DNS top-level labels for conformance with an alphabetic restriction will not recognize such corresponding A-Labels (i.e., U-labels represented in their US-ASCII form).

4. TLD Label Syntax Specification

This document relaxes the existing specification to allow TLD DNS-Labels to be well-formed A-Labels, but places restrictions on their corresponding U-Labels. That is, not every well-formed A-Label is a valid TLD DNS-Label.

The ABNF expression that matches a valid TLD DNS-Label is as follows:

tld-dns-label = traditional-tld-label / idn-label

traditional-tld-label = 1*63(ALPHA)

idn-label = Restricted-A-Label

ALPHA = %x41-5A / %x61-7A ; A-Z / a-z

A Restricted-A-Label is a DNS-Label which satisfies all the following conditions:

1. the DNS-Label is a valid A-Label according to [RFC5890];
2. the derived property value of all code points, as defined by [RFC5890], is PVALID;
3. the general category of all code points, is one of { Ll, Lo, Lm, Mn, Mc }.

This new specification reflects current practice in registration of TLD names by the IANA, extended to accommodate IDNs.

5. Policy Considerations

This document provides a technical specification that limits the set of TLD DNS-Labels that are available for assignment; it does not aim to encapsulate the full policy framework within which TLD names are chosen.

At the time of writing, the policy under which TLD names are chosen is developed and maintained by ICANN in consultation with a wide base of stakeholders. As the Internet continues to grow to serve new user communities, applications and services, it is to be expected that the corresponding policy will be changed accordingly.

6. IANA Considerations

While this document makes no requests of the IANA, management of the root zone is an IANA function. This document expands the set of strings permitted for delegation from the root zone, and hence establishes new limits for the corresponding IANA policy.

7. Security Considerations

This document is believed to have limited security implications.

General discussion about the security effects of internationalized labels can be found in [RFC5890], section 4. Those considerations apply equally to TLD labels.

The creation of new TLDs has the potential to conflict with software which (for example) predates and correspondingly does not accommodate new TLD names. Such software problems might in turn lead to security vulnerabilities, e.g. in the case where a DNS name specified by a user is truncated or otherwise misinterpreted, causing an application to interact with a different remote host from that which the user intended. It should be noted that this is not a new phenomenon, and has been observed following the creation of new (US-ASCII) TLD names prior to the publication of this document.

The issue that some Unicode characters can be confused with each other is discussed at length in the Security Considerations section of [RFC5890].

8. Acknowledgements

Tina Dam, Patrik Faltstrom, John Klensin, Thomas Narten and Andrew Sullivan contributed text to this document, and their contributions are hereby acknowledged.

9. References

9.1. Normative References

- [RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.
- [RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.
- [RFC5892] Faltstrom, P., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, August 2010.

9.2. Informative References

- [RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, October 1985.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

Appendix A. Change History

This section (and sub-sections) should be removed before publication.

A.1. draft-liman-tld-names-06

Add Mc as an allowable code-point, required for names in Devanagari script.

A.2. draft-liman-tld-names-05

New affiliation and address for Liman, due to company merger.

A.3. draft-liman-tld-names-04

Removed subjective and unverified statements regarding deployed software. Replaced with more generic text. Polishing a few expressions to make them less obtrusive. Removed confusing paragraph after ABNF table. Updated some references that are now published as RFCs.

A.4. draft-liman-tld-names-03

More wordsmithing, and explanatory text. Work on the IANA and the security considerations sections.

A.5. draft-liman-tld-names-02

Wordsmithing and rearrangement of text following discussions with Joe Abley, Tina Dam, Thomas Narten and Andrew Sullivan. Incorporated revised ABNF and associated specification from Patrik Faltstrom. Tightened definitions and introduced the term "DNS-Label" to avoid ambiguity with various other uses of the word "label".

A.6. draft-liman-tld-names-01

Substantial comments and improvements supplied by Thomas Narten and John Klensin. Decided to go for a minimal change approach. Also noted that U-labels have to be letters due to jumping digit problem. Rewritten major parts.

A.7. draft-liman-tld-names-00

First cut. Prompted by Olafur Gudmundsson and Tina Dam.

A.8. Version Identification Tag

\$Id: draft-liman-tld-names.xml,v 1.40 2011/04/12 08:20:42 liman Exp \$

Authors' Addresses

Lars-Johan Liman
Netnod Internet Exchange
Box 30194
SE-104 25 Stockholm
Sweden

Email: liman@netnod.se
URI: <http://www.netnod.se/>

Joe Abley
ICANN
4676 Admiralty Way
Suite 330
Marina del Rey 90292
USA

Email: joe.abley@icann.org
URI: <http://www.icann.org/>

Network Working Group
Internet-Draft
Updates: 2046 (if approved)
Intended status: Standards Track
Expires: January 12, 2012

A. Melnikov
Isode Limited
J. Reschke
greenbytes
July 11, 2011

Update to MIME regarding Charset Parameter Handling in Textual Media
Types
draft-melnikov-mime-default-charset-01

Abstract

This document changes RFC 2046 rules regarding default charset parameter values for text/* media types to better align with common usage by existing clients and servers.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 12, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1.	Introduction and overview	3
2.	Conventions Used in This Document	3
3.	New rules for default charset parameter values for text/* media types	3
4.	Default charset parameter value for text/plain media type	4
5.	Security Considerations	4
6.	IANA Considerations	4
7.	References	4
7.1.	Normative References	4
7.2.	Informative References	5
Appendix A.	Acknowledgements	5
	Authors' Addresses	5

1. Introduction and overview

[RFC2046] specified that the default charset parameter (i.e. the value used when it is not specified) is "US-ASCII". [RFC2616] changed the default for use by HTTP to be "ISO-8859-1". This encoding is not very common for new text/* media types and a special rule in HTTP adds confusion about which specification ([RFC2046] or [RFC2616]) is authoritative in regards to the default charset for text/* media types. [[anchor2: At the time of writing of this document the IETF HTTPBIS WG is working on an update to RFC 2616 which removes the default charset of "ISO-8859-1" for "text/*" media types. It is expected that the set of HTTPBIS documents will reference this document in order to use the updated rules of default charset in "text/*" media types.]]

Many complex text subtypes such as text/html [RFC2854] and text/xml [RFC3023] have internal (to their format) means of describing the charset. Many existing User Agents ignore the default of "US-ASCII" rule for at least text/html and text/xml.

This document changes RFC 2046 rules regarding default charset parameter values for text/* media types to better align with common usage by existing clients and servers.

2. Conventions Used in This Document

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

3. New rules for default charset parameter values for text/* media types

Section 4.1.2 of [RFC2046] says:

"The default character set, which must be assumed in the absence of a charset parameter, is US-ASCII."

As explained in the Introduction section this rule is considered to be outdated, so this document replaces it with the following set of rules:

Each subtype of the "text" media type which uses the "charset" parameter can define its own default value for the "charset" parameter, including absence of any default.

In order to improve interoperability with deployed agents, "text/*" media type definitions SHOULD either a) specify that the "charset" parameter is not used for the defined subtype, because the charset information is transported inside the payload (as in "text/xml") or b) require explicit unconditional inclusion of the "charset" parameter eliminating the need for a default value. In accordance with option (a), above, "text/*" media types that can transport charset information inside the corresponding payloads, specifically including "text/html" and "text/xml", SHOULD NOT specify the use of a "charset" parameter, nor any default value, in order to avoid conflicting interpretations should the charset parameter value and the value specified in the payload disagree.

New subtypes of the "text" media type, thus, SHOULD NOT define a default "charset" value. If there is a strong reason to do so despite this advice, they SHOULD use the "UTF-8" [RFC3629] charset as the default.

Specifications of how to specify the "charset" parameter, and what default value, if any, is used, are subtype-specific, NOT protocol-specific. Protocols that use MIME, therefore, MUST NOT override default charset values for "text/*" media types to be different for their specific protocol. The protocol definitions MUST leave that to the subtype definitions.

4. Default charset parameter value for text/plain media type

The default charset parameter value for text/plain is unchanged from [RFC2046] and remains as "US-ASCII".

5. Security Considerations

TBD. Guessing of default charset is a security problem. Conflicting information in-band vs out-of-band is also a security problem.

6. IANA Considerations

This document asks IANA to update the "text" subregistry of the Media Types registry to additionally point to this document.

7. References

7.1. Normative References

- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

7.2. Informative References

- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC2854] Connolly, D. and L. Masinter, "The 'text/html' Media Type", RFC 2854, June 2000.
- [RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

Appendix A. Acknowledgements

Many thanks to Ned Freed and John Klensin for comments and ideas that motivated creation of this document, and to Barry Leiba for suggested text.

Authors' Addresses

Alexey Melnikov
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com

Julian F. Reschke
greenbytes GmbH
Hafenweg 16
Muenster, NW 48155
Germany

Email: julian.reschke@greenbytes.de
URI: <http://greenbytes.de/tech/webdav/>

Applications Area Working Group
Internet-Draft
Intended status: Informational
Expires: January 1, 2012

J. Wang, Ed.
S. Ma
L. Liang
ZTE Corporation
June 30, 2011

Virtual Desktop Infrastructure Problem Statement
draft-wang-appsawg-vdi-problem-statement-01

Abstract

The Virtual Desktop Infrastructure is a technology to separate local desktop and remote computing/storage resources, which was initially derived from the remote desktop administration but with new business models and very different use cases. Most of existing VDI systems are based on proprietary implementation, and positioning different market with different features. Since virtual desktop technology is believed to be a mainstream application delivery method, like http protocol against web applications, so it's important to make the virtual desktop access protocol open and standard. This draft summarizes the limitations of existing virtual desktop systems, and proposes the intent standardization work in IETF.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 1, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
1.2. Terminology and Abbreviation	3
2. Virtual Desktop System Architecture	4
2.1. Common Framework	4
2.1.1. VDI server in the VMM	5
2.1.2. VDI Server In The Guest OS	6
2.2. Deployment System Architecture	6
2.3. VDI Protocol Stack Reference Architecture	8
3. Application scenarios	11
3.1. scenario 1: Enterprise IT Application	11
3.2. scenario 2: Cloud Hosted Virtual Computer	11
3.3. scenario 3: Cloud Hosted Telecommunication Terminals	12
3.4. scenario 4: Alternative SaaS delivery method	12
4. Known Issues	12
5. Intended Work In IETF	14
6. Acknowledgements	14
7. Security Considerations	15
8. References	15
8.1. Normative References	15
8.2. Informative References	15
Authors' Addresses	15

1. Introduction

The Virtual Desktop technology is a combination of remote desktop control and virtualization. Client can access a remote virtual desktop or a remote application through VDI protocol. Currently, there are a lot of VDI system vendors, including Citrix, Microsoft, VMware etc., each of them provides bundle of components based on proprietary implementation, it's difficult to interwork between different vendors. In addition, the existing solutions focus on enterprise scale application, we haven't found any one designed for public virtual desktop service offering.

Since the VDI will be an alternative application delivery method that is even more powerful than web, it can benefit a lot from a unified open and standard protocol, which ensures any qualified client can access any qualified server system. Thinking about WEB, without standardized http and other related protocols, its long-lasting prosperity could have been unimaginable.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Terminology and Abbreviation

- o OS: Operation System
- o Guest OS: An OS runs on top of a virtual machine manager.
- o VDI: Virtual Desktop Infrastructure.
- o VDI agent: software module running in Guest OS to perform special tasks, such as Clipboard sharing between Guest OS and VDI client. Specific functionality depends on VDI implementation.
- o VDI server: software module which implements VDI protocol stack and runs in the server side. Sometimes it is also called as Virtual Desktop Agent.
- o VDI client: software module which implements VDI protocol stack and runs in the client side. Sometimes it is also called as VDI viewer.
- o VM: Virtual Machine, contains at least one guest OS, usually several additional application processes.

- o VMM: Virtual Machine Manager.
- o VDA: Virtual Desktop Agent.

2. Virtual Desktop System Architecture

2.1. Common Framework

A VDI system includes at least one client and one server. VDI adopts the client-server model, in which the client runs a VDI viewer software which connects to the server through virtual desktop access protocol. Unlike the remote administration, VDI clients always connect to a dedicated virtual machine and don't share the desktop with others. The service provider or the enterprise's IT department must prepare exclusive VM image and virtual storage space for each client, and keep track of the clients' modification on desktop and storage.

Depend on the VDI system implementation, the VDI server module lies in either the VMM or the guest OS. Accordingly, the VDI systems can be divided into two categories.

2.1.1. VDI server in the VMM

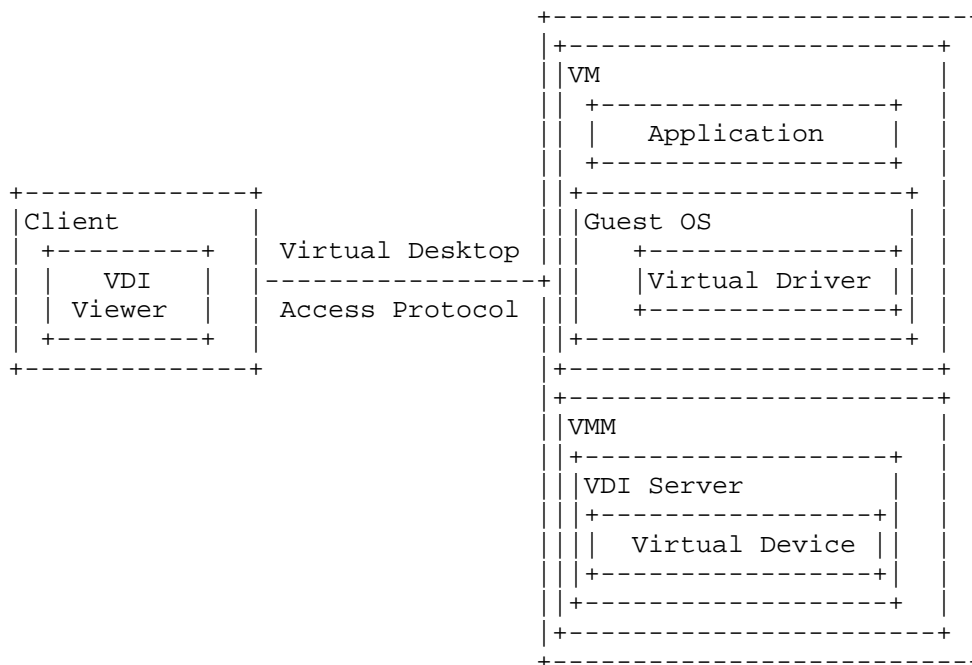


Figure 1 virtual desktop system framework: VDI server in the VMM

A VDI back-end system makes up of a VDI server module running inside the VMM and some virtual devices drivers. The VDI server is the virtual desktop access protocol processing endpoint on the server side, and also simulates several necessary virtual devices such as graphic card, usb devices and sound card. The corresponding drivers are installed into the guest OS, so the I/O operation can be pass-through along the application - guest OS - virtual driver-virtual device-VDI server - virtual desktop access protocol - VDI client path.

The different I/O information encapsulates into respective transport channel of the protocol, normally, including control, screen, mouse/keyboard input, audio input, audio output and usb devices channel. Any screen change of the server will be sent back to the client which is in charge of re-rendering the information to the client screen; similarly, the mouse, keyboard, audio input will be sent from client to the server and audio output, usb writing operation at the server sent to the client. All of these VDI system behaviors make sure that the user gets the same experience as operating his local computers

while accessing remote virtual desktop.

2.1.2. VDI Server In The Guest OS

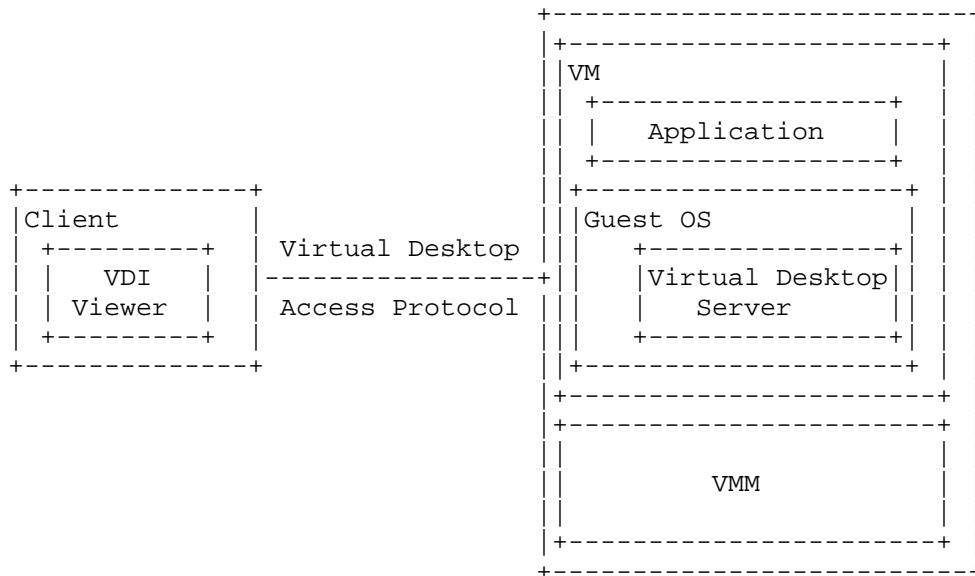


Figure 2 virtual desktop system framework- VDI server in the Guest OS

Compared to the first architecture, the only difference is virtual desktop server be installed in the guest OS. As a result, the connection terminates in the guest OS rather than in the VMM, so the client can't communicate with its virtual desktop until the guest OS finished startup.

Sometimes, The virtual desktop is also called 'virtual desktop agent', which performance the functions of both the virtual device and virtual driver modules, plus other necessary function.

2.2. Deployment System Architecture

A typical virtual desktop system comprises of clients, access gateways or connection brokers, authentication server, virtual desktop server pool and virtual application server, some additional component will also be deployed if extra features be required.

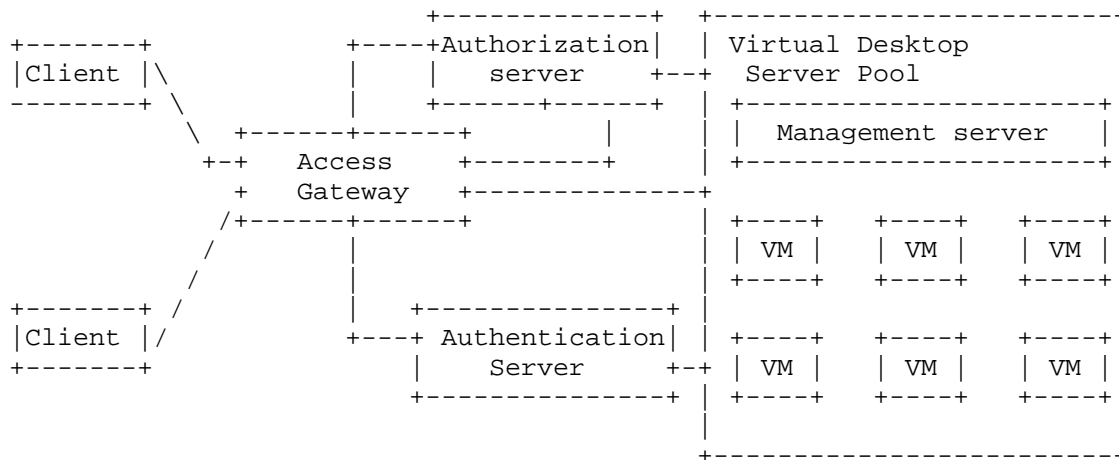


Figure 3 virtual desktop system deployment architecture

- o VDI Clients, usually run a virtual desktop client software, sometimes they're called viewer or receiver. The client software performances service discovery, resource connection and virtual desktop access through virtual desktop access protocol.
- o Access Gateway, which is a virtual desktop infrastructure entry point. If the client lies in the public network, the access gateway is used to authenticate the client and perform other security related function. In some specific VDI systems, the access gateway is replaced with redirector, which directs the traffic to a virtual desktop resource instead of forcing all of the traffic going through a designated gateway.
- o Authentication server, which stores required security information, responsible for authenticating the clients. Most of the VDI systems reuse existing facilities such the windows' Active Directory.
- o Authorization server, it contains the VDI subscriber's profile information, including the virtual desktop cpu number, memory and disk quota, application preference, etc. Sometimes, the authorization server is merged into authentication server, e.g., the active directory or Home subscriber server defined in 3GPP. In certain case, the authorization server may also act as resources allocating server, accepting the authorization request from the user or access gateways and checking if the user has the rights to access certain services. If the result is positive, the server sends a virtual resource allocating request to the resource

pool management server. After the virtual desktop resource successfully allocated, the user will be directed to connect the designated resource.

- o virtual desktop server pool, they're a bunch of computing and storage resources organized by virtualization technology. The Management server in charge of monitoring the VM state and allocating available resource to the user. Virtual machines are modified for desktop virtualization. Depending on the VDI business model, the scale of resources pool may vary from hundreds to millions virtual machine.

Not every vendor provides aforementioned components. Some of them ,doesn't implement access gateway, e.g., Redhat provides a web portal to redirect the authenticated client to the designated VM. Most of the vendors support application virtualization and application streaming, in which the application software are installed on some dedicated servers instead of keeping a copy on every virtual desktop machine, reducing the disk space requirement significantly and making the virtual machine mobility simpler.

Since the VDI systems intend to realize a 'virtual real desktop' for the client, besides the screen, mouse and keyboard input/output information transfer, most of them support the I/O redirection features, which means the remote virtual desktop can access the client's I/O devices, including audio and usb equipment, just like the devices have been installed at the remote servers.

2.3. VDI Protocol Stack Reference Architecture

Figure 4 provides one reference architecture for VDI protocol stack. From a high level view, it has Session Control Layer, Virtual Channel Layer and Application Layer above traditional Transport Layer. Each layer provides services to its upper layer while receiving services from the layer below. One thing needed to be mentioned here is that not all layers in this figure are in the scope of VDI protocol. VDI protocol will focus only on Session Control Layer and Virtual Channel Layer. For the other two layers, they are listed here for completeness and better understanding.

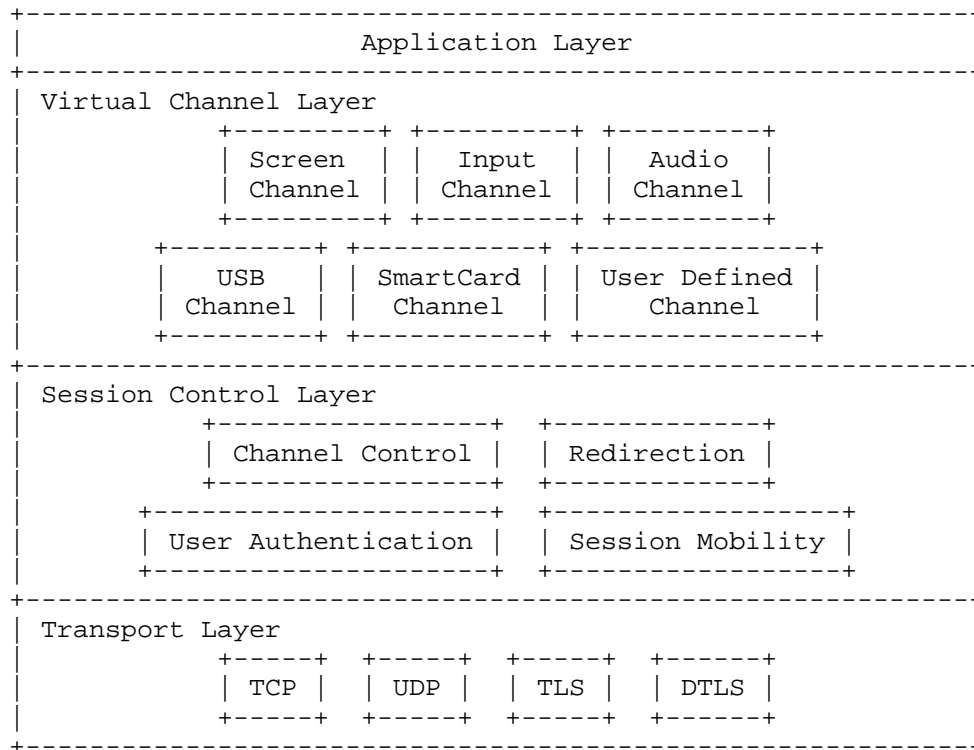


Figure 4 VDI protocol stack architecture

o Transport Layer

VDI protocol doesn't include transport protocol, but depends on the service provided by transport layer. It will rely on: 1) reliable connection-oriented service from transport layer to deliver control and sequential information, like graphics commands; 2) fast connection-less service from transport layer for delivering multimedia data; 3) security services from transport layer for data encryption, integrity and authentication. So any transport protocol that could satisfy above three requirements could be fit for VDI and they may be, but not limited, TCP/UDP/TLS/DTLS.

o Application Layer

Application Layer utilizes service from different VDI channels and provides VDI services to end-user. Its relationship with VDI protocol is somewhat like the relationship between Web browser (such as IE, Firefox) and HTTP protocol. It is also not in the scope of

VDI protocol.

- o Session Control Layer

Session Control Layer is mainly responsible for controlling the dialogues (connections) between VDI server and VDI client and general VDI services.

1. Channel Control: Functionalities of Channel Control include channel connection setup, deletion and maintenance. From implementation perspective, it will provide Service Access Point (SAP) and hook mechanism to virtual channel layer for these services. In this way, it isolates virtual channel layer from detailed connection management and provides foundation services for virtual channel plug-in. Also virtual channel layer will not deal with IP address directly, which makes session mobility easier.
2. User Authentication: User Authentication is responsible for authentication between VDI server and VDI client during connection setup. It provides one framework to exchange control information, for example algorithms negotiation. For detailed authentication algorithms, they are not included in the scope of VDI protocol.
3. Session Mobility: As indicated before, virtual channel layer will deal with unique name of end points, instead of network address, like IP. The mapping management between the unique name and IP address is done in Session Mobility. Whenever IP address of end-point is changed due to handoff between different networks (for example, between LTE and WiFi), it will notify this change to the other end-point. Channel Control will re-establish all the connection, which is transparent to virtual channel layer. In this way, continuous service is guaranteed.
4. Redirection: Redirection is responsible for user redirection. When the user first accesses the VDI server, the server may redirect the user to another VDI server because the virtual desktop has been migrated away. This functionality could be used in scenarios like load balance, user roaming scenarios and etc.

- o Virtual Channel Layer

Different information of VDI desktop is encapsulated into different channels. Adopting this multi-channel architecture is due to different service requirements of different channels, such as security, priority (QoS) and etc. Basic channels include:

1. Screen Channel: This channel is responsible for delivering desktop graphics information from VDI server to VDI client. The content in the message may include: graphics commands, bitmaps, cache ID and etc.
2. Input Channel: This channel is responsible for exchanging input information between VDI server and VDI client. VDI client sends keyboard/mouse event to the server, while the server transfers cursor shape and pointer position to the client.
3. Audio Channel: This channel is responsible for exchanging audio information between VDI server and VDI client, including audio playing and recording.
4. USB Channel: This channel is responsible for redirecting USB devices which plugs in the VDI client to the VDI server, so that user could operate USB devices in the VDI desktop.
5. Smart Card Channel: This channel is responsible for redirecting Smart Card devices which plugs in the VDI client to the VDI server, so that Smart Card based authentication could be performed.

Besides above channels, user could provide channels with specific functions as the plug-in. This is achieved through utilizing Channel Control services.

3. Application scenarios

3.1. scenario 1: Enterprise IT Application

The enterprise built a centralized virtual desktop resource pool, where each of the office staffs has been designated at least a VM. The employees run the office software and access the enterprise application through their virtual desktop. Any devices, e.g., thin clients, laptops, tablets and smartphones, can be used as virtual desktop clients without extra cross-platform software porting costs. Storing sensitive data at local computer is restricted so that information leakage risk can be minimized.

3.2. scenario 2: Cloud Hosted Virtual Computer

Virtual Computer service providers provide VM to their customers who access their VM via virtual desktop protocol. The subscription fees depend on the contracted computing resources, such as CPU number, memory capacity, disk quota, and their usage statistics.

The subscribers can access their virtual computer via any devices and any media, the benefit for them includes: 1) Reduction of the devices hardware upgrading period, thus hardware cost can be cut down. 2) Keeping data, documents synchronized among their devices. 3) Installing software once and using them anywhere and on any devices.

3.3. scenario 3: Cloud Hosted Telecommunication Terminals

The carriers supply the virtual desktop host service to their customers, which allows the clients to use the telecommunication application similar to local phones, such as voice/video calls, messages and address book, and furthermore, keep the same phone number as their real phones.

The benefits of this kind of business model are: 1) Multi-screen service convergence, which means the users can access the same application and content via different screens and get the same experience, and can migrate the services among different devices in realtime. 2) Get rid of the terminals' hardware limitations, one can even play a complex 3D game on an ordinary smartphone. Of course, the smart phone only acts as an I/O device while the game is running on the remote VM. 3) Accelerating the time-to-market period of new telecom services. Formerly, if the new application impacts on the terminals' software, the operator must persuade the users to upgrade their devices, this would spend very long time. But if the terminals were hosted in the carrier's network, the upgrading can be done centralized in a short time.

3.4. scenario 4: Alternative SaaS delivery method

Till now, most of the SaaS is delivered by WEB technology, but it's still a hard task to realize some sophisticated applications, such as graphical design and 3D game, in web pages. These cons of WEB are just the pros of virtual desktop. By using virtual desktop, SaaS can be extended to any software products.

4. Known Issues

At the very beginning, the virtual desktop solution targeted the enterprise market, so the office software support is a priority. Therefore, most VDI products, especially the Microsoft's RDS/RDP product family, are highly optimized for the office software. Our test results show that RDP session only produces tens of kilobytes per second traffic while running Microsoft's Word or Excel software with regular document reading or editing operation, as well as the Internet surfing.

Some of the VDI vendors claimed that their systems have been optimized for multimedia applications, however, the test results reveal that they are not fully optimized as their assertion especially under low-speed access environment. We have tested the streaming application performance at the platform based on Redhat's spice and Microsoft's RDP protocol. The spice protocol chooses M-JPEG as the streaming data compression algorithm, and the test results disclose that it has obvious better performance compared to Microsoft's RDP+terminal services combination. Despite the optimization efforts done by Redhat, the spice still consumes 12 times access bandwidth compared to the original bitrates of a standard definition video, that's totally not acceptable while accessing the services via carrier's wireless network.

M-JPEG does not use inter-frame encoding, which reduces the processing overhead but results in lower compression ratio. MPEG-4/h.264's compression ratio is about 5-10 times compared better than M-JPEG, but requires more powerful CPU or GPU hardware. Recently, even the low end desktop or laptop systems have been equipped with integrated GPU capable of HD MPEG-4 video decoding. At the mobile sector, most of the 3G mobile phones have the h.264 decoding chipset installed, and the high-end handsets have shipped with an integrated GPU, e.g., iPhone 4/iPad.

Secondly, the audio streaming overhead also need to be cut down. The widely adopted audio codec in mobile network is AMR or it's wideband version - the AMR-WB, cost bandwidth about 6.6 to 23.85 kilobits per second, if encoded within rtp, the bitrate ranges between 23kbps and 40kbps with 20ms ptime parameter. In contrast, the regular audio streaming in the VDI system consumes 128kps or higher bandwidth. the excess overhead can not be neglected in the mobile network.

Thirdly, assuming that the providers use VDI technology to deliver services, they purchase server hardware and software from different vendors for some commercial considerations, but still hope to provide the service to any potential customers whatever the device they used to access the network. So all of the participants would benefit from the fact that any VDI client device running any vendor's software can connect to the VDI backend services built by any vendor's solution, that is, a unified open and standard protocol should be widely adopted. It is true also for migrating virtual machines across different providers. Furthermore, due to the VDI backend system variance, the access security mechanism may totally different, e.g., Active Directory vs. 3gpp sim based access security, so it'd better to separate the security framework from the basic VDI protocol design.

Fourthly, there're more and more mobile devices connecting to the

network via wireless technology, the mobility management and service continuity should be a fundamental capability of the VDI protocol. Someone may argue that the mobility is the network's business, but actually, network based mobility can't provide inter-network ip mobility at most cases, we still need keep eye on the mobility issues while VDI client switches between a 3G network to another carrier's WiFi hotspot.

5. Intended Work In IETF

- o Designing a fundamental virtual desktop access protocol, which enables the interworking between different VDI clients and backends. The protocol should support screen, mouse and keyboard mapping between clients and remote systems as well as audio and usb devices redirection.
- o Defining a framework to accommodate advanced video and audio compression algorithm, especially allow the client negotiates intended algorithm with backend system.
- o Defining an access security framework, in which the client and server can negotiate preferred security suite. To meet the minimal interworking requirement, some access security mechanism must be defined as a mandatory component, e.g., Pre-shared Key or PKI method, some other authentication mechanism also can be introduced, e.g., the 3gpp SIM based security. If the integrated security also need to be achieved, the protocol should support transfer credential information from clients to servers, e.g., restricted access to the client's certificates or SIM card from virtual desktop server side.
- o Integrating the mobility and service continuity capabilities. To facilitate the user roaming among different network providers, mobility and service continuity should not rely on the underlying network facilities. Another service continuity related issue is caused by the VM live migration in the backend system, the protocol design should also take it into account.

6. Acknowledgements

Many members from different societies and companies provided useful comments and feedback on this document. Thanks for their efforts in reviewing the document and providing valuable comments.

7. Security Considerations

Related security issues will be addressed in subsequent draft.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4867] Sjöberg, J., Westerlund, M., Lankaniemi, A., and Q. Xie, "RTP Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs", RFC 4867, April 2007.
- [spice] redhat, spice project., "Spice remote computing protocol definition vl.0", 2009.

8.2. Informative References

- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

Authors' Addresses

Jun Wang (editor)
ZTE Corporation
No.68 Zijinghua Road
Nanjing,
CN

Phone: +86 13770604455
Email: wang.jun17@zte.com.cn

Suan Ma
ZTE Corporation
No.68 Zijinghua Road
Nanjing,
CN

Phone:
Email: ma.suan@zte.com.cn

Liang Liang
ZTE Corporation
No.68 Zijinghua Road
Nanjing,
CN

Phone:
Email: liang.liang12@zte.com.cn

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 4, 2013

F. Galiegue, Ed.
K. Zyp, Ed.
SitePen (USA)
G. Court
January 31, 2013

JSON Schema: core definitions and terminology
draft-zyp-json-schema-04

Abstract

JSON Schema defines the media type "application/schema+json", a JSON based format for defining the structure of JSON data. JSON Schema provides a contract for what JSON data is required for a given application and how to interact with it. JSON Schema is intended to define validation, documentation, hyperlink navigation, and interaction control of JSON data.

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 August 4, 2013.

Copyright Notice

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

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

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions and Terminology	3
3. Core terminology	3
3.1. Property, item	3
3.2. JSON Schema, keywords	3
3.3. Empty schema	3
3.4. Root schema, subschema	4
3.5. JSON Schema primitive types	4
3.6. JSON value equality	5
3.7. Instance	5
4. Overview	5
4.1. Validation	5
4.2. Hypermedia and linking	6
5. General considerations	6
5.1. Applicability to all JSON values	6
5.2. Programming language independence	6
5.3. JSON Schema and HTTP	6
5.4. JSON Schema and other protocols	6
5.5. Mathematical integers	7
5.6. Extending JSON Schema	7
5.7. Security considerations	7
6. The "\$schema" keyword	7
6.1. Purpose	7
6.2. Customization	8
7. URI resolution scopes and dereferencing	8
7.1. Definition	8
7.2. URI resolution scope alteration with the "id" keyword	8
7.2.1. Valid values	8
7.2.2. Usage	9
7.2.3. Canonical dereferencing and inline dereferencing	10
7.2.4. Inline dereferencing and fragments	11
7.3. Interoperability considerations	11
8. Recommended correlation mechanisms for use with the HTTP protocol	11
8.1. Correlation by means of the "Content-Type" header	11
8.2. Correlation by means of the "Link" header	12
9. IANA Considerations	12
10. References	12
10.1. Normative References	12
10.2. Informative References	12
Appendix A. ChangeLog	13

1. Introduction

JSON Schema is a JSON media type for defining the structure of JSON data. JSON Schema provides a contract for what JSON data is required for a given application and how to interact with it. JSON Schema is intended to define validation, documentation, hyperlink navigation, and interaction control of JSON data.

This specification defines JSON Schema core terminology and mechanisms; related specifications build upon this specification and define different applications of JSON Schema.

2. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

The terms "JSON", "JSON text", "JSON value", "member", "element", "object", "array", "number", "string", "boolean", "true", "false", and "null" in this document are to be interpreted as defined in RFC 4627 [RFC4627].

3. Core terminology

3.1. Property, item

When referring to a JSON Object, as defined by [RFC4627], the terms "member" and "property" may be used interchangeably.

When referring to a JSON Array, as defined by [RFC4627], the terms "element" and "item" may be used interchangeably.

3.2. JSON Schema, keywords

A JSON Schema is a JSON document, and that document MUST be an object. Object members (or properties) defined by JSON Schema (this specification, or related specifications) are called keywords, or schema keywords.

A JSON Schema MAY contain properties which are not schema keywords.

3.3. Empty schema

An empty schema is a JSON Schema with no properties, or with properties which are not schema keywords.

3.4. Root schema, subschema

This example of a JSON Schema has no subschemas:

```
{
  "title": "root"
}
```

JSON Schemas can also be nested, as in this example:

```
{
  "title": "root",
  "otherSchema": {
    "title": "nested",
    "anotherSchema": {
      "title": "alsoNested"
    }
  }
}
```

In this example, "nested" and "alsoNested" are subschemas, and "root" is a root schema.

3.5. JSON Schema primitive types

JSON Schema defines seven primitive types for JSON values:

array A JSON array.

boolean A JSON boolean.

integer A JSON number without a fraction or exponent part.

number Any JSON number. Number includes integer.

null The JSON null value.

object A JSON object.

string A JSON string.

3.6. JSON value equality

Two JSON values are said to be equal if and only if:

- both are nulls; or

- both are booleans, and have the same value; or

- both are strings, and have the same value; or

- both are numbers, and have the same mathematical value; or

- both are arrays, and:

 - have the same number of items; and

 - items at the same index are equal according to this definition;
or

- both are objects, and:

 - have the same set of property names; and

 - values for a same property name are equal according to this definition.

3.7. Instance

An instance is any JSON value. An instance may be described by one or more schemas.

An instance may also be referred to as "JSON instance", or "JSON data".

4. Overview

This document proposes a new media type "application/schema+json" to identify JSON Schema for describing JSON data. JSON Schemas are themselves written in JSON. This, and related specifications, define keywords allowing to describe this data in terms of allowable values, textual descriptions and interpreting relations with other resources. The following sections are a summary of features defined by related specifications.

4.1. Validation

JSON Schema allows applications to validate instances, either non interactively or interactively. For instance, an application may

collect JSON data and check that this data matches a given set of constraints; another application may use a JSON Schema to build an interactive interface in order to collect user input according to constraints described by JSON Schema.

4.2. Hypermedia and linking

JSON Schema provides a method for extracting link relations from instances to other resources, as well as describing interpretations of instances as multimedia data. This allows JSON data to be interpreted as rich hypermedia documents, placed in the context of a larger set of related resources.

5. General considerations

5.1. Applicability to all JSON values

It is acknowledged that an instance may be any valid JSON value as defined by [RFC4627]. As such, JSON Schema does not mandate that an instance be of a particular type: JSON Schema can describe any JSON value, including null.

5.2. Programming language independence

JSON Schema is programming language agnostic. The only limitations are the ones expressed by [RFC4627] and those of the host programming language.

5.3. JSON Schema and HTTP

This specification acknowledges the role of HTTP [RFC2616] as the dominant protocol in use on the Internet, and the wealth of official specifications related to it.

This specification uses a subset of these specifications to recommend a set of mechanisms, usable by this protocol, to associate JSON instances to one or more schemas.

5.4. JSON Schema and other protocols

JSON Schema does not define any semantics for the client-server interface for any other protocols than HTTP. These semantics are application dependent, or subject to agreement between the parties involved in the use of JSON Schema for their own needs.

5.5. Mathematical integers

It is acknowledged by this specification that some programming languages, and their associated parsers, use different internal representations for floating point numbers and integers, while others do not.

As a consequence, for interoperability reasons, JSON values used in the context of JSON Schema, whether that JSON be a JSON Schema or an instance, SHOULD ensure that mathematical integers be represented as integers as defined by this specification.

5.6. Extending JSON Schema

Implementations MAY choose to define additional keywords to JSON Schema. Save for explicit agreement, schema authors SHALL NOT expect these additional keywords to be supported by peer implementations. Implementations SHOULD ignore keywords they do not support.

5.7. Security considerations

Both schemas and instances are JSON values. As such, all security considerations defined in RFC 4627 [RFC4627] apply.

6. The "\$schema" keyword

6.1. Purpose

The "\$schema" keyword is both used as a JSON Schema version identifier and the location of a resource which is itself a JSON Schema, which describes any schema written for this particular version.

This keyword MUST be located at the root of a JSON Schema. The value of this keyword MUST be a URI [RFC3986] and a valid JSON Reference [json-reference]; this URI MUST be both absolute and normalized. The resource located at this URI MUST successfully describe itself. It is RECOMMENDED that schema authors include this keyword in their schemas.

The following values are predefined:

<http://json-schema.org/schema#> JSON Schema written against the current version of the specification.

<http://json-schema.org/hyper-schema#> JSON Schema written against the current version of the specification.

<http://json-schema.org/draft-04/schema#> JSON Schema written against this version.

<http://json-schema.org/draft-04/hyper-schema#> JSON Schema hyperschema written against this version.

<http://json-schema.org/draft-03/schema#> JSON Schema written against JSON Schema, draft v3 [json-schema-03].

<http://json-schema.org/draft-03/hyper-schema#> JSON Schema hyperschema written against JSON Schema, draft v3 [json-schema-03].

6.2. Customization

When extending JSON Schema with custom keywords, schema authors SHOULD define a custom URI for "\$schema". This custom URI MUST NOT be one of the predefined values.

7. URI resolution scopes and dereferencing

7.1. Definition

JSON Schema uses JSON Reference [json-reference] as a mechanism for schema addressing. It extends this specification in two ways:

- JSON Schema offers facilities to alter the base URI against which a reference must resolve by the means of the "id" keyword;

- it defines a specific dereferencing mechanism extending JSON Reference to accept arbitrary fragment parts.

Altering the URI within a schema is called defining a new resolution scope. The initial resolution scope of a schema is the URI of the schema itself, if any, or the empty URI if the schema was not loaded from a URI.

7.2. URI resolution scope alteration with the "id" keyword

7.2.1. Valid values

The value for this keyword MUST be a string, and MUST be a valid URI. This URI MUST be normalized, and SHOULD NOT be an empty fragment (#) or the empty URI.

7.2.2. Usage

The "id" keyword (or "id", for short) is used to alter the resolution scope. When an id is encountered, an implementation **MUST** resolve this id against the most immediate parent scope. The resolved URI will be the new resolution scope for this subschema and all its children, until another id is encountered.

When using "id" to alter resolution scopes, schema authors **SHOULD** ensure that resolution scopes are unique within the schema.

This schema will be taken as an example:

```
{
  "id": "http://x.y.z/rootschema.json#",
  "schema1": {
    "id": "#foo"
  },
  "schema2": {
    "id": "otherschema.json",
    "nested": {
      "id": "#bar"
    },
    "alsonested": {
      "id": "t/inner.json#a"
    }
  },
  "schema3": {
    "id": "some://where.else/completely#"
  }
}
```

Subschemas at the following URI-encoded JSON Pointer [json-pointer]s (starting from the root schema) define the following resolution scopes:

```
# (document root) http://x.y.z/rootschema.json#
#/schema1 http://x.y.z/rootschema.json#foo
#/schema2 http://x.y.z/otherschema.json#
#/schema2/nested http://x.y.z/otherschema.json#bar
```



```
#/schema2/alsoNested http://x.y.z/t/inner.json#a
```

```
#/schema3 some://where.else/completely#
```

7.2.3. Canonical dereferencing and inline dereferencing

When resolving a URI against a resolution scope, an implementation may choose two modes of operation:

canonical dereferencing The implementation dereferences all resolved URIs.

inline dereferencing The implementation chooses to dereference URIs within the schema.

Implementations **MUST** support canonical dereferencing, and **MAY** support inline dereferencing.

For example, consider this schema:

```
{
  "id": "http://my.site/myschema#",
  "definitions": {
    "schema1": {
      "id": "schema1",
      "type": "integer"
    },
    "schema2": {
      "type": "array",
      "items": { "$ref": "schema1" }
    }
  }
}
```

When an implementation encounters the "schema1" reference, it resolves it against the most immediate parent scope, leading to URI "http://my.site/schema1#". The way to process this URI will differ according to the chosen dereferencing mode:

if canonical dereferencing is used, the implementation will dereference this URI, and fetch the content at this URI;

if inline dereferencing is used, the implementation will notice that URI scope "http://my.site/schema1#" is already defined within the schema, and choose to use the appropriate subschema.

7.2.4. Inline dereferencing and fragments

When using inline dereferencing, a resolution scope may lead to a URI which has a non empty fragment part which is not a JSON Pointer, as in this example:

```
{
  "id": "http://some.site/schema#",
  "not": { "$ref": "#inner" },
  "definitions": {
    "schema1": {
      "id": "#inner",
      "type": "boolean"
    }
  }
}
```

An implementation choosing to support inline dereferencing SHOULD be able to use this kind of reference. Implementations choosing to use canonical dereferencing, however, are not required to support it.

7.3. Interoperability considerations

Inline dereferencing can produce canonical URIs which differ from the canonical URI of the root schema. Schema authors SHOULD ensure that implementations using canonical dereferencing obtain the same content as implementations using inline dereferencing.

Extended JSON References using fragments which are not JSON Pointers are not dereferenceable by implementations choosing not to support inline dereferencing. This kind of reference is defined for backwards compatibility, and SHOULD NOT be used in new schemas.

8. Recommended correlation mechanisms for use with the HTTP protocol

It is acknowledged by this specification that the majority of interactive JSON Schema processing will be over HTTP. This section therefore gives recommendations for materializing an instance/schema correlation using mechanisms currently available for this protocol. An instance is said to be described by one (or more) schema(s).

8.1. Correlation by means of the "Content-Type" header

It is RECOMMENDED that a MIME type parameter by the name of "profile" be appended to the "Content-Type" header of the instance being processed. If present, the value of this parameter MUST be a valid

URI, and this URI SHOULD resolve to a valid JSON Schema. The MIME type MUST be "application/json", or any other subtype.

An example of such a header would be:

```
Content-Type: application/my-media-type+json;
             profile=http://example.com/my-hyper-schema#
```

8.2. Correlation by means of the "Link" header

When using the "Link" header, the relation type used MUST be "describedBy", as defined by RFC 5988, section 5.3 [RFC5988]. The target URI of the "Link" header MUST be a valid JSON Schema.

An example of such a header would be:

```
Link: <http://example.com/my-hyper-schema#>; rel="describedBy"
```

9. IANA Considerations

The proposed MIME media type for JSON Schema is defined as follows:

```
type name: application;
subtype name: schema+json.
```

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.
- [RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010.
- [json-reference] Bryan, P. and K. Zyp, "JSON Reference (work in progress)", September 2012, <<http://tools.ietf.org/html/draft-pbryan-zyp-json-ref-03>>.
- [json-pointer] Bryan, P. and K. Zyp, "JSON Pointer (work in progress)", September 2012, <<http://tools.ietf.org/html/draft-ietf-appsawg-json-pointer-07>>.
- [json-schema-03] Court, G. and K. Zyp, "JSON Schema, draft 3", September 2012, <<http://tools.ietf.org/html/draft-zyp-json-schema-03>>.

Appendix A. ChangeLog

draft-00

- * Initial draft.
- * Salvaged from draft v3.
- * Mandate the use of JSON Reference, JSON Pointer.
- * Define the role of "id". Define URI resolution scope.
- * Add interoperability considerations.

Authors' Addresses

Francis Galiegue (editor)

EMail: fgaliegue@gmail.com

Kris Zyp (editor)
SitePen (USA)
530 Lytton Avenue
Palo Alto, CA 94301
USA

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

Gary Court
Calgary, AB
Canada

EMail: gary.court@gmail.com

