Abstract

Section 1.1.1 of RFC 3986 defines URI syntax as "a federated and extensible naming system wherein each scheme’s specification may further restrict the syntax and semantics of identifiers using that scheme." In other words, the structure of a URI is defined by its scheme. While it is common for schemes to further delegate their substructure to the URI’s owner, publishing independent standards that mandate particular forms of substructure in URIs is often problematic.

This document provides guidance on the specification of URI substructure in standards.

Note to Readers

_RFC EDITOR: please remove this section and the "URIs" section before the Appendices before publication_

This is a proposed revision of RFC7320, aka BCP190. The -00 draft is a copy of the published RFC; subsequent revisions will update it.

The issues list for this draft can be found at https://github.com/mnot/I-D/labels/rfc7320bis [1].

The most recent (often, unpublished) draft is at https://mnot.github.io/I-D/rfc7320bis/ [2].

Recent changes are listed at https://github.com/mnot/I-D/commits/gh-pages/rfc7320bis [3].

See also the draft’s current status in the IETF datatracker, at https://datatracker.ietf.org/doc/draft-nottingham-rfc7320bis/ [4].
Internet-Draft          URI Design and Ownership            January 2020

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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on July 9, 2020.

Copyright Notice

Copyright (c) 2020 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 (https://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.  Intended Audience ..................................... 4
    1.2.  Notational Conventions ................................. 5
2.  Best Current Practices for Standardizing Structured URIs .... 5
    2.1.  URI Schemes ........................................... 5
    2.2.  URI Authorities ....................................... 5
    2.3.  URI Paths ............................................. 6
    2.4.  URI Queries .......................................... 6
    2.5.  URI Fragment Identifiers ............................... 7
3.  Alternatives to Specifying Structure in URIs ................. 7
4.  Security Considerations ...................................... 8
5.  IANA Considerations ........................................ 8
6.  References .................................................. 8
    6.1.  Normative References .................................. 8

Nottingham                Expires July 9, 2020                  [Page 2]
1. Introduction

URIs [RFC3986] very often include structured application data. This might include artifacts from filesystems (often occurring in the path component) and user information (often in the query component). In some cases, there can even be application-specific data in the authority component (e.g., some applications are spread across several hostnames to enable a form of partitioning or dispatch).

Implementations can impose further constraints upon the structure of URIs; for example, many Web servers use the filename extension of the last path segment to determine the media type of the response. Likewise, prepackaged applications often have highly structured URIs that can only be changed in limited ways (often, just the hostname and port on which they are deployed).

Because the owner of the URI (as defined in [webarch] Section 2.2.2.1) is choosing to use the server or the application, this can be seen as reasonable delegation of authority. However, when such conventions are mandated by a party other than the owner, it can have several potentially detrimental effects:

- Collisions - As more ad hoc conventions for URI structure become standardized, it becomes more likely that there will be collisions between them (especially considering that servers, applications, and individual deployments will have their own conventions).

- Dilution - When the information added to a URI is ephemeral, this dilutes its utility by reducing its stability (see [webarch] Section 3.5.1), and can cause several alternate forms of the URI to exist (see [webarch] Section 2.3.1).

- Rigidity - Fixed URI syntax often interferes with desired deployment patterns. For example, if an authority wishes to offer several applications on a single hostname, it becomes difficult to impossible to do if their URIs do not allow the required flexibility.

- Operational Difficulty - Supporting some URI conventions can be difficult in some implementations. For example, specifying that a particular query parameter be used with "HTTP" URIs can preclude the use of Web servers that serve the response from a filesystem.
Likewise, an application that fixes a base path for its operation (e.g., "/v1") makes it impossible to deploy other applications with the same prefix on the same host.

- Client Assumptions - When conventions are standardized, some clients will inevitably assume that the standards are in use when those conventions are seen. This can lead to interoperability problems; for example, if a specification documents that the "sig" URI query parameter indicates that its payload is a cryptographic signature for the URI, it can lead to undesirable behavior.

Publishing a standard that constrains an existing URI structure in ways that aren't explicitly allowed by [RFC3986] (usually, by updating the URI scheme definition) is therefore sometimes problematic, both for these reasons, and because the structure of a URI needs to be firmly under the control of its owner.

This document explains some best current practices for establishing URI structures, conventions, and formats in standards. It also offers strategies for specifications in Section 3.

1.1. Intended Audience

This document’s guidelines and requirements target the authors of specifications that constrain the syntax or structure of URIs or parts of them. Two classes of such specifications are called out specifically:

- Protocol Extensions ("Extensions") - specifications that offer new capabilities that could apply to any identifier, or to a large subset of possible identifiers; e.g., a new signature mechanism for 'http' URIs, metadata for any URI, or a new format.

- Applications Using URIs ("Applications") - specifications that use URIs to meet specific needs; e.g., an HTTP interface to particular information on a host.

Requirements that target the generic class "Specifications" apply to all specifications, including both those enumerated above and others.

Note that this specification ought not be interpreted as preventing the allocation of control of URIs by parties that legitimately own them, or have delegated that ownership; for example, a specification might legitimately define the semantics of a URI on IANA’s Web site as part of the establishment of a registry.

There may be existing IETF specifications that already deviate from the guidance in this document. In these cases, it is up to the
relevant communities (i.e., those of the URI scheme as well as that which produced the specification in question) to determine an appropriate outcome; e.g., updating the scheme definition, or changing the specification.

1.2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] when, and only when, they appear in all capitals, as shown here.

2. Best Current Practices for Standardizing Structured URIs

This section updates [RFC3986] by advising Specifications how they should define structure and semantics within URIs. Best practices differ depending on the URI component in question, as described below.

2.1. URI Schemes

Applications and Extensions can require use of specific URI scheme(s); for example, it is perfectly acceptable to require that an Application support 'http' and 'https' URIs. However, Applications ought not preclude the use of other URI schemes in the future, unless they are clearly only usable with the nominated schemes.

A Specification that defines substructure for URI schemes overall (e.g., a prefix or suffix for URI scheme names) MUST do so by modifying [BCP35] (an exceptional circumstance).

2.2. URI Authorities

Scheme definitions define the presence, format and semantics of an authority component in URIs; all other Specifications MUST NOT constrain, or define the structure or the semantics for URI authorities, unless they update the scheme registration itself, or the structures it relies upon (e.g., DNS name syntax, defined in Section 3.5 of [RFC1034]).

For example, an Extension or Application cannot say that the "foo" prefix in "http://foo_app.example.com" is meaningful or triggers special handling in URIs, unless they update either the HTTP URI scheme, or the DNS hostname syntax.
Applications can nominate or constrain the port they use, when applicable. For example, BarApp could run over port nnnn (provided that it is properly registered).

2.3. URI Paths

Scheme definitions define the presence, format, and semantics of a path component in URIs, although these are often delegated to the application(s) in a given deployment.

To avoid collisions, rigidity, and erroneous client assumptions, Specifications MUST NOT define a fixed prefix for their URI paths; for example, "/myapp", unless allowed by the scheme definition.

One such exception to this requirement is registered "well-known" URIs, as specified by [RFC8615]. See that document for a description of the applicability of that mechanism.

Note that this does not apply to Applications defining a structure of URIs paths "under" a resource controlled by the server. Because the prefix is under control of the party deploying the application, collisions and rigidity are avoided, and the risk of erroneous client assumptions is reduced.

For example, an Application might define "app_root" as a deployment-controlled URI prefix. Application-defined resources might then be assumed to be present at "{app_root}/foo" and "{app_root}/bar".

Extensions MUST NOT define a structure within individual URI components (e.g., a prefix or suffix), again to avoid collisions and erroneous client assumptions.

2.4. URI Queries

The presence, format and semantics of the query component of URIs is dependent upon many factors, and can be constrained by a scheme definition. Often, they are determined by the implementation of a resource itself.

Applications can specify the syntax of queries for the resources under their control. However, doing so can cause operational difficulties for deployments that do not support a particular form of a query. For example, a site may wish to support an Application using "static" files that do not support query parameters.

Extensions MUST NOT constrain the format or semantics of queries, to avoid collisions and erroneous client assumptions. For example, an Extension that indicates that all query parameters with the name
"sig" indicate a cryptographic signature would collide with potentially preexisting query parameters on sites and lead clients to assume that any matching query parameter is a signature.

HTML [W3C.REC-html401-19991224] constrains the syntax of query strings used in form submission. New form languages are encouraged to allow creation of a broader variety of URIs (e.g., by allowing the form to create new path components, and so forth).

2.5. URI Fragment Identifiers

Section 3.5 of [RFC3986] specifies fragment identifiers’ syntax and semantics as being dependent upon the media type of a potentially retrieved resource. As a result, other Specifications MUST NOT define structure within the fragment identifier, unless they are explicitly defining one for reuse by media types in their definitions (for example, as JSON Pointer [RFC6901] does).

An Application that defines common fragment identifiers across media types not controlled by it would engender interoperability problems with handlers for those media types (because the new, non-standard syntax is not expected).

3. Alternatives to Specifying Structure in URIs

Given the issues described in Section 1, the most successful strategy for Applications and Extensions that wish to use URIs is to use them in the fashion they were designed: as links that are exchanged as part of the protocol, rather than statically specified syntax. Several existing specifications can aid in this.

[RFC8288] specifies relation types for Web links. By providing a framework for linking on the Web, where every link has a relation type, context and target, it allows Applications to define a link’s semantics and connectivity.

[RFC6570] provides a standard syntax for URI Templates that can be used to dynamically insert Application-specific variables into a URI to enable such Applications while avoiding impinging upon URI owners’ control of them.

[RFC8615] allows specific paths to be ‘reserved’ for standard use on URI schemes that opt into that mechanism (‘http’ and ‘https’ by default). Note, however, that this is not a general "escape valve" for applications that need structured URIs; see that specification for more information.
Specifying more elaborate structures in an attempt to avoid collisions is not an acceptable solution, and does not address the issues in Section 1. For example, prefixing query parameters with "myapp_" does not help, because the prefix itself is subject to the risk of collision (since it is not "reserved").

4. Security Considerations

This document does not introduce new protocol artifacts with security considerations. It prohibits some practices that might lead to vulnerabilities; for example, if a security-sensitive mechanism is introduced by assuming that a URI path component or query string has a particular meaning, false positives might be encountered (due to sites that already use the chosen string). See also [RFC6943].

5. IANA Considerations

There are no direct IANA actions specified in this document.

6. References

6.1. Normative References


6.2. Informative References

6.3. URIs


Appendix A. Changes from RFC7320

Many of the requirements of RFC7320 were removed, in the spirit of making this BCP guidance, rather than rules.
Appendix B. Acknowledgments

Thanks to David Booth, Dave Crocker, Tim Bray, Anne van Kesteren, Martin Thomson, Erik Wilde, Dave Thaler and Barry Leiba for their suggestions and feedback.

Author’s Address

Mark Nottingham

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