Internet Engineering Task Force (IETF) M. Nottingham
Request for Comments: 7320
BCP: 190 July 2014
Updates: 3986
Category: Best Current Practice
ISSN: 2070-1721

URI Design and Ownership

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 URI substructure is inappropriate, because that essentially usurps ownership. This document further describes this problematic practice and provides some acceptable alternatives for use in standards.

Status of This Memo

This memo documents an Internet Best Current Practice.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on BCPs is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7320.

Table of Contents

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).

Furthermore, constraints upon the structure of URIs can be imposed by an implementation; 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 precludes 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 inappropriate, because the structure of a URI needs to be firmly under the control of its owner, and the IETF (as well as other organizations) should not usurp it.

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

1.1. Intended Audience

This document's 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, or metadata for any URI.

- 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", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Best Current Practices for Standardizing Structured URIs

This section updates [RFC3986] by setting limitations on how other specifications may define structure and semantics within URIs. Best practices differ depending on the URI component, as described below.

2.1. URI Schemes

Applications and extensions MAY 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 SHOULD 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 within a specific URI scheme MUST do so in the defining document for that URI scheme. A specification that defines substructure for URI schemes overall MUST do so by modifying [BCP115] (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.

For example, an extension or application ought not say that the "foo" prefix in "foo_app.example.com" is meaningful or triggers special handling in URIs.

However, applications MAY 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; all other specifications MUST NOT constrain, or define the structure or the semantics for any path component.

The only exception to this requirement is registered "well-known" URIs, as specified by [RFC5785]. See that document for a description of the applicability of that mechanism.

For example, an application ought not specify a fixed URI path "/myapp", since this usurps the host's control of that space.

Specifying a fixed path relative to another (e.g., {whatever}/myapp) is also bad practice (even if "whatever" is discovered as suggested in Section 3); while doing so might prevent collisions, it does not avoid the potential for operational difficulties (for example, an implementation that prefers to use query processing instead, because of implementation constraints).

2.4. URI Queries

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

Applications MUST NOT directly specify the syntax of queries, as this 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.

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 SHOULD NOT emulate it, but instead allow creation of a broader variety of URIs (e.g., by allowing the form to create new path components, and so forth).

Note that "well-known" URIs (see [RFC5785]) MAY constrain their own query syntax, since these name spaces are effectively delegated to the registering party.

2.5. URI Fragment Identifiers

Media type definitions (as per [RFC6838]) SHOULD specify the fragment identifier syntax(es) to be used with them; other specifications MUST NOT define structure within the fragment identifier, unless they are explicitly defining one for reuse by media type definitions.

For example, 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.

[RFC5988] 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.

[RFC5785] 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. References

5.1. Normative References

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

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

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[webarch] Jacobs, I. and N. Walsh, "Architecture of the World Wide Web, Volume One", December 2004, <http://www.w3.org/TR/2004/REC-webarch-20041215>.

5.2. Informative References

[BCP115] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", RFC 4395, BCP 115, February 2006.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, March 2012.

[RFC6943] Thaler, D., "Issues in Identifier Comparison for Security Purposes", RFC 6943, May 2013.

[W3C.REC-html401-19991224] Raggett, D., Hors, A., and I. Jacobs, "HTML 4.01 Specification", World Wide Web Consortium Recommendation REC-html401-19991224, December 1999, <http://www.w3.org/TR/1999/REC-html401-19991224>.

Appendix A. 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: http://www.mnot.net/