Thing-to-Thing Research Group
Internet-Draft
Intended status: Informational
Expires: February 19, 2016

K. Hartke
Universitaet Bremen TZI

August 18, 2015

CoRE Application Descriptions
draft-hartke-core-apps-02

Abstract

The interfaces of RESTful, hypertext-driven applications consist of reusable, self-descriptive components such as Internet media types and link relation types. This document defines a template that application designers can use to describe their application's interface in a structured way so that other parties can develop interoperable clients and servers or reuse the components in their own applications.

Note to Readers

This Internet-Draft should be discussed on the Thing-to-Thing Research Group (T2TRG) mailing list <t2trg@irtf.org>.

Status of This Memo

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

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

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

This Internet-Draft will expire on February 19, 2016.

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] is designed to enable applications implementing the REST architectural style [REST] in Constrained-Node Networks [RFC7228].

As CoAP applications are implemented and deployed, it becomes increasingly important to be able to describe them in some structured way in order to promote interoperability and reuse. Previous efforts like WADL [WADL] however focus on code generation from machine-readable service descriptions and are not truly RESTful [DWNWADL].

REST application interfaces (APIs) are by definition hypertext-driven [RESTAPI]. This means that the interface is a description of the common vocabulary between the client and the server, centered around Internet media types and link relation types, rather than a static list of resources and the operations supported on them.

RESTful applications are often easy to understand, but require some design effort. This is because application designers do not only have to take current requirements into consideration, but also anticipate changes that may be required in the future. The reward is long-term stability and evolvability ("design for decades"). REST is intended for long-lived network-based applications that span multiple organizations [RESTAPI].

This document defines a template for describing the interface of RESTful, hypertext-driven applications in Constrained RESTful Environments (CoRE).

2. Application Descriptions

A CoRE Application Description is a named set of reusable, self-descriptive components. It is comprised of:

- URI schemes that identify communication protocols,
- Internet media types that identify representation formats,
- link relation types that identify link semantics,
- form relation types that identify form semantics, and
- optionally, well-known locations.

Together, these components provide the specific, in-band instructions for interfacing with a given service.

2.1. Communication Protocols

The foundation of a hypertext-driven REST API are the communication protocol(s) spoken between a client and a server. Although HTTP/1.1 [RFC7230] is by far the most common communication protocol for REST APIs, a REST API should typically not be dependent on any specific communication protocol.

2.2. URI Schemes

The use of a particular protocol is guided by URI schemes [RFC7595] that describe the syntax and semantics of URI references found in links and forms (Section 2.4).

A URI scheme refers to a family of protocols, typically distinguished by a version number. For example, the "http" URI scheme refers to the two members of the HTTP family of protocols: HTTP/1.1 [RFC7230], and HTTP/2 [RFC7540]. The specific HTTP version is negotiated

between the client and the server through version indicators in the protocol or the TLS application-layer protocol negotiation (ALPN) extension [RFC7301].

IANA maintains a list of registered URI schemes at <http://www.iana.org/assignments/uri-schemes>.

2.3. Internet Media Types

One of the most important aspect of hypertext-driven communications is the concept of media types [RFC6838]. Media types are used to label representations so that it is known how the representation should be interpreted, and how it is encoded. The core of an application description should be one or more hypertext media types.

A media type identifies a versioned series of representation formats (Section 2.4): a media type does not identify a particular version of a representation format; rather, the media type identifies the family, and includes provisions for version indicator(s) embedded in the representations themselves to determine more precisely the nature of how the data is to be interpreted. A new media type is only needed to designate a completely incompatible format [MIMEWEB].

Media types consist of a top-level type and a subtype, structured into trees. Optionally, media types can have parameters. For example, the media type "text/plain; charset=utf-8" is a subtype for generic text under the "text" top-level type in the standards tree, and has a parameter "charset" set to "utf-8".

Media types can be further refined by structured type name suffixes (e.g., "+xml" appended to the base subtype name; see Section 4.2.8 of RFC 6838), or by subtype information embedded in the representations themselves (e.g., "xmlns" declarations in XML documents [XMLNS]). Structured type name suffixes should be preferred, because embedded subtype information cannot be negotiated (e.g., using the CoAP Accept option).

A media type must be determined from in-band information (e.g., from the CoAP Content-Format option). Clients must not assume a structure from the application context or other out-of-band information.

IANA maintains a list of registered Internet media types at <http://www.iana.org/assignments/media-types>.

IANA maintains a list of registered structured suffixes at <http://www.iana.org/assignments/media-type-structured-suffix>.

IANA maintains a list of registered CoAP content formats at <http://www.iana.org/assignments/core-parameters>.

2.4. Representation Formats

In RESTful applications, clients and servers exchange representations that capture the current or intended state of a resource and that are labeled with a media type. A representation is a sequence of bytes whose structure and semantics are specified by a representation format, a set of rules for encoding information.

Representation formats should generally allow clients with different goals, so they can do different things with the same data. The specification of a representation format "describes a problem space, not a prescribed relationship between client and server. Client and server must share an understanding of the representations they're passing back and forth, but they don't need to have the same idea of what the problem is that needs to be solved." [WEBAPIS]

Representation formats and their specifications evolve over time. It is part of the responsibility of the designer of a new version of a format to try to insure both forward and backward compatibility: new documents should work reasonably (with some fallback) with old processors, and old documents should work reasonably with new processors [MIMEWEB].

Representation formats enable hypertext-driven applications when they support the expression of hypermedia controls: links (Section 2.4.1) and/or forms (Section 2.4.2).

2.4.1. Links

A link is the primary means for a client to change application state, i.e., to navigate from one resource to another. A link is a typed connection between two resources [RFC5988], and is comprised of:

- a context (usually the current resource),
- a link relation type that identifies the semantics of the link (Section 2.5),
- a target resource URI, and
- optionally, attributes that describe the link target.

A link can be viewed as a statement of the form "{context IRI} has a {relation type} resource at {target URI}, which has {target attributes}" [RFC5988]. For example, <http://example.com/> might

have a "terms-of-service" resource at <http://example.com/tos>, which has the media type "text/html".

There are two special kind of links:

- An embedding link is a link with the additional hint that it, when processed, should be substituted with a representation of the referenced resource. Thus, traversing an embedding link adds to the application state, rather than replacing it.

- A templated link is a link where the client constructs the target resource URI from provided in-band instructions. The specific rules for such instructions are described by the representation format. URI Templates [RFC6570] provide a generic way to construct URIs through variable expansion.

2.4.2. Forms

A form is the primary means for a client to change resource state. It is comprised of:

- a context (usually the current resource),

- a form relation type that identifies the semantics of the form (Section 2.6),

- a target resource URI,

- a submission method (PUT, POST, PATCH, or DELETE), and

- and a description of a representation that the service accepts as part of form submission. This description can be a set of form fields, or simply a list of acceptable media types.

A form can be viewed as an instruction of the form "To {relation type} {context}, make a {method} request to {target URI}". For example, to "create-item" in <http://example.com/collection/>, a client might make a POST request to <http://example.com/collection/>.

Note: A form with a submission method of GET is, strictly speaking, a templated link, since it provides a way to construct a URI and does not change resource state.

2.5. Link Relation Types

A link relation type identifies the semantics of a link [RFC5988]. For example, a link with the relation type "copyright" indicates that

the resource identified by the target URI is a statement of the copyright terms applying to the current context.

Relation types are not to be confused with media types [RFC6838]; they do not identify the format of the representation that results when the link is dereferenced. Rather, they only describe how the current context is related to another resource.

IANA maintains a list of registered link relation types at <http://www.iana.org/assignments/link-relations>.

2.6. Form Relation Types

A form relation type identifies the semantics of a form. For example, a form with the relation type "create-item" indicates that a new item can be created within the current context by making a request to the resource identified by the target URI.

IANA maintains a list of registered link relation types at <TBD>.

2.7. Well-Known Locations

Some applications may require the discovery of information about a host ("site-wide metadata"). For example, [RFC6415] defines a metadata document format for describing hosts; similarly, [RFC6690] defines a link format for the discovery of resources hosted by a server.

Applications that need to define a resource for site-wide metadata can register new "well-known locations". [RFC5785] defines a path prefix in "http" and "https" URIs for this purpose, "/.well-known/"; [RFC7252] extends this concept to "coap" and "coaps" URIs.

IANA maintains a list of registered well-known URIs at <http://www.iana.org/assignments/well-known-uris>.

2.8. URI Structures

Application descriptions must not constrain URI structures in ways that aren't explicitly allowed by [RFC3986]. In particular, mandating particular forms of URI substructure is inappropriate. [RFC7320] describes this problematic practice and provides some acceptable alternatives for use in application descriptions.

3. Template

Application name:

URI schemes:

Media types:

Link relations:

Form relations:

Well-known locations:

Interoperability considerations:

Security considerations:

Contact:

Author/Change controller:

4. Security Considerations

The security considerations of [RFC3986], [RFC5785], [RFC5988], [RFC6570], [RFC6838], [RFC7320], and [RFC7595] are inherited.

All components of an application description are expected to contain clear security considerations. Application descriptions should further contain security considerations that need to be taken into account for the security of the overall application.

5. IANA Considerations

5.1. Form Relation Type Registry

This specification establishes the Form Relation Type registry.

5.1.1. Registering New Form Relation Types

Form relation types are registered in the same way as link relation types [RFC5988], i.e., they are registered on the advice of a Designated Expert, with a Specification Required.

The requirements for registered relation types are adopted from [RFC5988], Section 4.1.

The registration template is:

- Relation Name:
- Description:
- Reference:
- Notes: [optional]

5.1.2. Initial Registry Contents

The Form Relation Type registry's initial contents are:

- Relation Name: create-item
  Description: Refers to a resource that can be used to create a resource in a collection of resources.
  Reference: [RFCXXXX]

- Relation Name: delete
  Description: Refers to a resource that can be used to delete a resource in a collection of resources.
  Reference: [RFCXXXX]

- Relation Name: update
  Description: Refers to a resource that can be used to update the state of the form's context.
  Reference: [RFCXXXX]

6. Acknowledgements

Thanks to Jan Algermissen, Mike Amundsen, Olaf Bergmann, Carsten Bormann, Stefanie Gerdes, Mike Kelly, Michael Koster, Matthias Kovatsch, Julian Reschke, Teemu Savolainen, Bilhanan Silverajan, and Erik Wilde for helpful comments and discussions that have shaped the document.

Some of the text in this document has been borrowed from [RESTAPI], [RFC5988], [RFC7320], and [MIMEWEB]. All errors are my own.

This work was funded in part by Nokia.

7. References

7.1. Normative References

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

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <http://www.rfc-editor.org/info/rfc5785>.

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

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <http://www.rfc-editor.org/info/rfc6570>.

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

[RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, DOI 10.17487/RFC7320, July 2014, <http://www.rfc-editor.org/info/rfc7320>.

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <http://www.rfc-editor.org/info/rfc7595>.

7.2. Informative References

[DWNWADL] Gregorio, J., "Do we need WADL?", June 2007, <http://bitworking.org/news/193/Do-we-need-WADL>.

[MIMEWEB] Masinter, L., "MIME and the Web", draft-masinter-mime-web-info-02 (work in progress), January 2011.

[REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. Dissertation, University of California, Irvine, 2000, <http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>.

[RESTAPI] Fielding, R., "REST APIs must be hypertext-driven", October 2008, <http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven>.

[RFC6415] Hammer-Lahav, E., Ed. and B. Cook, "Web Host Metadata", RFC 6415, DOI 10.17487/RFC6415, October 2011, <http://www.rfc-editor.org/info/rfc6415>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <http://www.rfc-editor.org/info/rfc6690>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <http://www.rfc-editor.org/info/rfc7228>.

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

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <http://www.rfc-editor.org/info/rfc7301>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[WADL] Hadley, M., "Web Application Description Language", World Wide Web Consortium Member Submission SUBM-wadl-20090831, August 2009, <http://www.w3.org/Submission/2009/SUBM-wadl-20090831>.

[WEBAPIS] Richardson, L. and M. Amundsen, "RESTful Web APIs", O'Reilly, September 2013.

[XMLNS] Bray, T., Hollander, D., Layman, A., Tobin, R., and H. Thompson, "Namespaces in XML 1.0 (Third Edition)", World Wide Web Consortium Recommendation REC-xml-names-20091208, December 2009, <http://www.w3.org/TR/2009/REC-xml-names-20091208>.

Author's Address

Klaus Hartke
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63905
EMail: hartke@tzi.org