Thing-to-Thing Research Group
Internet-Draft
Intended status: Informational
Expires: 4 September 2025

C. Bormann
Universit辰t Bremen TZI
C. Ams端ss

3 March 2025

The "dereferenceable identifier" pattern
draft-bormann-t2trg-deref-id-05

Abstract

In a protocol or an application environment, it is often important to be able to create unambiguous identifiers for some meaning (concept or some entity).

Due to the simplicity of creating URIs, these have become popular for this purpose. Beyond the purpose of identifiers to be uniquely associated with a meaning, some of these URIs are in principle _dereferenceable_, so something can be placed that can be retrieved when encountering such a URI.

// The present revision -04 includes a few clarifications.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-bormann-t2trg-deref-id/.

Discussion of this document takes place on the t2trg Research Group mailing list (mailto:t2trg@irtf.org), which is archived at https://mailarchive.ietf.org/arch/browse/t2trg/. Subscribe at https://www.ietf.org/mailman/listinfo/t2trg/.

Source for this draft and an issue tracker can be found at https://github.com/cabo/deref-id.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

(Please see abstract.)

1.1. Terminology

Identifier

Dereferenceable

Dereference

Information: The information that is retrieved by dereferencing a dereferenceable identifier.

Operator: Dereferencing requires some server infrastructure to actually provide the _information_. Simplifying the potential complexity of this infrastructure, the entity (entities) controlling the operation of this server infrastructure, including the name spaces in use (e.g., DNS names, URI paths on a server) are called the operator(s) of the dereferenceable identifier.

Consumer: An entity that receives data containing a dereferenceable identifier.

Directed: A directed identifier is an identifier that has been specifically minted to not just identify the intended entity, but also context information such as the intended use, or intended consumer of the identifier.

Directed _information_ is _information_ that is tailored to the implicit context of a specific dereferencing access, such as the accessing IP address or other ancillary parameters. (Content negotiation alone is not "directed _information_", as it is explicitly triggered by the dereferencing entity.)

Unique: A unique identifier is an identifier that is unique for the entity; i.e., no other identifiers are in use (or intended to be in use).

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 [BCP14] (RFC2119) (RFC8174) when, and only when, they appear in all capitals, as shown here.

2. Examples for "dereferenceable identifiers"

This section is intended to present a number of examples where dereferenceable identifiers are in use in a protocol, including existing discussion about constraints on their usage, the benefits claimed for this constrained usage, and remaining issues.

2.1. Protocol and Protocol Version identifiers

Many protocols based on XML or JSON include a protocol or protocol version identifier in the heading to a data item.

E.g., [JSO] defines a language for data models that contain an identifier to the language version in use, here https://json-schema.org/draft/2020-12/schema. The model that can be retrieved from this URI in turn contains further dereferenceable identifiers that point to further details.

Section 8.1.1 of [JSO] has this:

> If this URI identifies a retrievable resource, that resource SHOULD be of media type "application/schema+json".

So it acknowledges that the dereferenceability is optional, but does place further restrictions on what can be the result of a successful dereference: another one of these data models, which in turn contain further dereferenceable identifiers.

2.2. Concept identifiers

The _Problem Details for HTTP APIs_ format [PROBLEM] uses a dereferenceable identifier for its "type" field. The value is a URI that "identifies the specific "problem type" (e.g., "out of credit")" (Section 1 of [PROBLEM]).

Section 3.1.1 of [PROBLEM] has this:

> If the type URI is a locator (e.g., those with an "http" or "https" scheme), dereferencing it SHOULD provide human-readable documentation for the problem type (e.g., using HTML [HTML5]).

but then warns:

> However, consumers SHOULD NOT automatically dereference the type URI, unless they do so when providing information to developers (e.g., when a debugging tool is in use).

Section 4 of [PROBLEM] further details:

> A problem type URI SHOULD resolve to HTML [HTML5] documentation that explains how to resolve the problem.

This becomes even more interesting as Section 4.2 of [PROBLEM] then gives this advice:

> | Registrations MAY use the prefix "https://iana.org/assignments/
> | http-problem-types#" for the type URI.

A reference to the place where registrations for these items are managed is certainly desirable, however, the implications on the management of fragment identifiers in the HTML documents that IANA generates from registration information are an example for the increased complexity dereferenceable identifiers may place on the owners of the URI space employed.

2.3. MORE EXAMPLES

There are a lot more examples in published RFCs; add them to this document.

3. Pitfalls

3.1. Server overload

If a data item containing dereferenceable identifier(s) becomes widely distributed, naive implementations that handle such a data item might dereference these identifiers as part of a routine operation. Many definitions of dereferenceable identifiers contain admonitions that such a behavior can cause an implosion of requests on the server(s) for the URI.

3.2. Longevity of identifiers

Dereferenceable URIs usually contain domain names, whose ownership can change. As a result, and for other reasons as well, parts of the name space of an origin may come under new administration, which can change the policies that apply to resources made available there.

These are problems of such URIs in general (and can be mitigated by going to a non-dereferenceable kind of URIs such as one based on the 'tag' uri scheme [TAG]). However, the problems are exacerbated by their use as a dereferenceable identifier. The new owner/ administrator might more easily accept that a certain chunk of their URI space should not be used (which suffices for a non-dereferenceable identifier based on this kind of URI namespace) than that certain content needs to be offered there (potentially presenting non-trivial loads, some mechanisms needed to update that information, and legal liabilities that are hard to assess).

3.3. Breakage due to incompatible changes

Dereferencing an identifier may produce different representations over time. While these changes may be intentional and beneficial (e.g. because terms are compatibly added to a resource describing terms that are evolved together [COOL]), they can also cause breakage in applications that previously dereferenced the identifier successfully:

* There can be errors in the representation introduced by the change.

* The operator and the consumer may disagree about what constitutes a compatible change.

* An updated representation may exceed the consumer’s capabilities, e.g. not fitting in an allocated buffer.

* Even without intended changes to the representation, changes to the channel may exclude certain consumers. For example, the operator’s web server may cease to accept the cipher suites implemented in the consumer.

* When the operator’s services are compromised, there may be malicious changes in the representation.

3.4. Redirect ambiguities

Dereferencing an identifier may involve following some redirections; whether that following is actually implied, or desired (or even desirable) is rarely being discussed.

3.5. Other pitfalls

Denial of service attacks are discussed in Section 6. Privacy implications, in particular around single-use identifiers, are discussed in Section 7.

4. Usage patterns between dereferencing and precise matching

Consumers may choose to:

* dereference a dereferenceable identifier and, in place of the dereferenceable identifier, using only the information retrieved this way, or

* treat the dereferenceable identifier as opaque.

Consumers do not face a binary choice between either; the space between those extremes is continuous.

Notable steps consumers can take to mitigate pitfalls of dereferencing are:

1. Consumers that dereference may apply caching, which reduces server load and bridges both outages and misconfigurations on the server side.

   These caches may adhere to the caching rules of the underlying systems (DNS result life times, HTTP's freshness rules), but may also stretch them if the alternative are failures or treating the identifier as opaque.

2. Consumers may use caching proxy services provided by trusted parties.

   While this may have an impact on the susceptibility to service outages, it immediately mitigates the privacy implications of having the consumer's network address visible to the operator. Restrictive policies at the proxy can further mitigate other issues. For example, if the proxy's cache is eagerly populated by web spider operations from public starting points and only ever serves cached results to consumers, it defends against single-use URIs.

3. Consumer caches may be pre-populated as part of their firmware update mechanisms.

   In its extreme form, the consumer may not even be equipped to dereference any identifiers outside of its cache, and the dereferenced representation may already be part of the firmware in ingested form to save runtime resources. Such a consumer shares its properties with a consumer that treats dereferenceable identifiers as opaque. However, the authors of the firmware can make good use of the dereferenceable identifiers. For example, they can dereference a known (or spidered) set of identifiers in an automated fashion, with any suitable amount of caching or manual verification.

5. IANA Considerations

This document makes no concrete requests on IANA, but does point out that IANA resources might be a good target for a certain class of dereferenceable identifiers.

6. Security considerations

The ability to create a denial of service attack by pointing a dereferenceable identifier into a popular data item that is widely distributed is implied by the discussion in Section 2, alongside with some recommendations for implementers that would mitigate such attacks. A problem with such recommendations is that they need to be followed by implementations that are using dereferenceable identifiers, which might not care much.

7. Privacy considerations

Dereferencing an identifier leaves a wide-spread data trail, ranging from host name lookups visible on the network to the absolute URI (i.e., the URI without its fragment identifier) visible to the operator of the identifier. Moreover, the operator might gain additional data about the requester, e.g. from a User-Agent header.

By minting directed (e.g., single-use) dereferenceable identifiers and assigning short cache lifetimes to the dereferenced resource, the originator of a document can track dereferencing clients whenever they process the document the identifier has been created for. Moreover, single-use identifiers can also be used to exfiltrate data from originators whose network access is restricted through dereferencing clients.

8. References

8.1. Normative References

[BCP14] Best Current Practice 14, <https://www.rfc-editor.org/info/bcp14>. At the time of writing, this BCP comprises the following:

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

[COOL] Sauermann, L. and R. Cyganiak, "Cool URIs for the Semantic Web", 3 December 2008, <https://www.w3.org/TR/cooluris/>.

[HTML5] WHATWG, "HTML — Living Standard", n.d., <https://html.spec.whatwg.org>.

[JSO] Wright, A., Andrews, H., Hutton, B., and G. Dennis, "JSON Schema: A Media Type for Describing JSON Documents", Work in Progress, Internet-Draft, draft-bhutton-json-schema-01, 10 June 2022, <https://datatracker.ietf.org/doc/html/draft-bhutton-json-schema-01>.

[PROBLEM] Nottingham, M., Wilde, E., and S. Dalal, "Problem Details for HTTP APIs", RFC 9457, DOI 10.17487/RFC9457, July 2023, <https://www.rfc-editor.org/rfc/rfc9457>.

[TAG] Kindberg, T. and S. Hawke, "The 'tag' URI Scheme", RFC 4151, DOI 10.17487/RFC4151, October 2005, <https://www.rfc-editor.org/rfc/rfc4151>.

Acknowledgements

Christian Amsss pointed out that this document would be good to have.

Authors' Addresses

Carsten Bormann
Universitt Bremen TZI
Postfach 330440
D-28359 Bremen
Germany
Phone: +49-421-218-63921
Email: cabo@tzi.org

Christian Amsss
Austria
Email: christian@amsuess.com