Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 5, 2013

M. Nottingham
Rackspace

July 4, 2012

Indicating Details of Problems to Machines in HTTP
draft-nottingham-http-problem-00

Abstract

This specification proposes a "Problem Detail" as an extensible way to carry machine-readable details of HTTP errors in a response, to avoid the need to invent new response formats for non-human consumers.

Note to Readers

This specification is tentative (as all Internet-Drafts are, but even more so). It's not yet clear whether defining this format will improve things enough to balance out the harm it may cause, and so it may disappear at any time. Using it before it becomes an RFC may subject you to ridicule; you have been warned.

What's the problem? In a nutshell, this sort of thing might encourage tighter coupling, making applications more brittle. On the other hand, lots of people are doing this already, so having a common way to do it -- with lots of caveats about how it can be misused -- might be better. Stay tuned.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Requirements 3
3. The Problem Details JSON Object 4
4. Expressing Problem Details with Link Headers 5
5. Defining New Problem Details 5
6. Generating Problem Details 7
7. Consuming Problem Details 7
8. Security Considerations 8
9. IANA Considerations 8
10. Acknowledgements 9
11. Normative References 9
Author's Address 9

1. Introduction

While HTTP [RFC2616] defines the status code as the primary indicator of response semantics, it is sometimes not fine-grained enough to convey helpful information about the error, particularly to non-human consumers.

For example, consider a 403 Forbidden response that indicates that the client's account doesn't have enough credit. While this can be adequately expressed in HTML if it's presented to a human in front of a Web browser, a non-browser client will have difficulty understanding the response, because it doesn't understand the structure of the markup.

This specification defines a "Problem Detail" as an extensible way to carry machine-readable details of errors in a response, to avoid the need to invent new response formats.

Problem details have:

- A generic status, identified and solely conveyed by the HTTP status code <http://www.iana.org/assignments/http-status-codes/>. For example, "403 Forbidden".
- An problem type that refines the status code semantics, identified by an absolute URI [RFC3986] and paired with a summary title.
- Optionally, detail that is specific to an instance of the problem.

The common data model for problem details as a JSON [RFC4627] object. That object can be serialised as an "application/json-problem" HTTP response, or it can be conveyed using a Link HTTP response header field [RFC5988].

Future work may include defining a HTML Microformat <http://microformats.org/> that allows this information to be carried in an HTML document as well.

Note that Problem Details are not the only way to convey the details of a problem in HTTP; if the response is still a representation of a resource, for example, it's often preferable to accommodate describing the details in that format.

Rather, the aim of this specification is to define a common format for those applications that need one, so that they aren't required to define their own.

2. Requirements

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. The Problem Details JSON Object

The canonical format for problem details is a JSON [RFC4627] document, identified with the "application/json-problem" media type. Its root object has the following properties:

- "describedby" (string) - A URL that identifies the type of problem. When dereferenced, it SHOULD provide human-readable documentation (e.g., using HTML).
- "title" (string) - A short, human-readable summary of the problem. It SHOULD NOT change from instance to instance of the problem, except for purposes of localisation.
- "detail" (string) - Optionally, additional, human readable detail specific to this instance of the problem.

Problem details, when serialised as JSON objects, MUST include the "describedby" and "title" properties. The "detail" property is always OPTIONAL.

Problem types MAY extend their instances with additional properties. For example:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json-problem
Content-Language: en

{
  "describedby": "http://example.com/probs/out-of-credit",
  "title": "You do not have enough credits.",
  "detail": "Your current balance is 30, but that costs 50.",
  "balance": 30,
  "account": "http://api.example.com/account/12345"
}
```

Here, the out-of-credit problem (identified by its URI) indicates the reason for the 403 in "title", gives instance-specific details in "detail", and adds two extensions; "balance" conveys the account's balance, and "account" gives a link where the account can be topped up.

Note that "describedby" is case-sensitive in the JSON object, as are all other property names.

4. Expressing Problem Details with Link Headers

Because resources often use formats other than JSON to convey human-readable problem details, it is also possible to serialise Problem Details for non-browser consumption alongside them using a Link response header field [RFC5988]. For example:

```
HTTP/1.1 403 Forbidden
Content-Type: text/html
Content-Language: en
Link: <http://example.com/probs/out-of-credit>; rel="describedby";
         title="You do not have enough credits."; balance=30;
         account="http://api.example.com/account/12345"
```

Here, the same problem as above is conveyed using the header in previous examples, but without the optional detail.

Problem details are serialised into Link headers using the "describedby" link relation type, with the URL for the problem type as the link target. The title is conveyed using the "title" parameter, and the detail can be conveyed using the "detail" parameter.

Extension parameters on the JSON Object can also be expressed as target parameters on the link header, as long as the extension either defines them as a string, or explains how to express them as one.

The title, detail, and extension properties MUST use the encoding specified in [RFC5987] if it falls outside of the US-ASCII character set.

For example, the following are all valid:

```
Link: <http://example.com/p/1>; rel="describedby"; title="Hi"
Link: <ftp://example.com/p/1>; rel="describedby"; title="Hi"
Link: <http://example.com/p/1>; rel="describedby";
      title="Hi there"; detail="something happened."
Link: <http://example.com/p/1>; rel="describedby"; other="foo";
      title="Hi there"; title*=UTF-8''Hi%20there; another=bar;
      detail*=UTF-8''something%20happened
```

5. Defining New Problem Details

Before defining a new type of problem detail, it's important to understand what they are good for, and what's better left to other mechanisms.

Problem details are not a debugging tool for the underlying implementation; rather, they are a way to expose greater detail about the HTTP interface of the application itself.

For example, an "out of credit" problem is an appropriate explanation as to why a request was forbidden. In contrast, a description of the internal conditions that led to a 500 Internal Server Error along with a stack trace are not an appropriate use of this mechanism, because it exposes implementation detail, rather than explaining the interface.

At the other end of the spectrum, truly generic problems - i.e., conditions that could potentially apply to any resource on the Web - are usually better expressed as plain status codes.

For example, a "write access disallowed" problem is unnecessary, since a 403 Forbidden status code on a PUT request is self-explanatory.

Finally, a problem domain may have a more appropriate way to carry an error in a format that it already defines. Problem details are intended to avoid the necessity of establishing new "fault" or "error" document formats, not to replace existing domain-specific formats. That said, it is possible to add support for problems using HTTP server-driven content negotiation (i.e., the client uses the Accept request header to indicate a preference for problems).

If defining a new problem still seems wise, one can be created by:

1. Nominating a describedby URL (typically, with the "http" scheme),
2. Choosing a title that appropriately describes it (think short), and
3. Nominating a HTTP status code for it to be used with.

Problem types MAY specify the use of the Retry-After response header in appropriate circumstances.

A problem's describedby URL SHOULD resolve to HTML documentation that explains how to resolve the problem.

Optionally, a problem definition MAY specify additional properties on the Problem Details JSON object.

For example, an extension might use typed links [RFC5988] to another resource that can be used by machines to resolve the problem.

If an extension is defined, its name SHOULD conform to token [RFC2616], so that it can be serialised in header and other formats.

Likewise, problems defining extensions should either make their values strings, or explain how to map their values to strings that are safe to include in HTTP headers (noting the special semantics of a comma there).

6. Generating Problem Details

The URL used in the describedby property MUST NOT be changed in any way from that defined by its specification; doing so turns it into a different problem that most consumers will not recognise.

The title string SHOULD be sent as specified by the problem definition, although it MAY be localised (e.g., using server-driven content negotiation, with the "Accept-Language" request header). Remember that such responses are required by HTTP to include a Vary header, e.g.:

```
HTTP/1.1 403 Forbidden
Content-Type: text/html
Content-Language: se
Vary: Accept-Language
Link: <http://example.com/probs/out-of-credit>;
      rel="describedby";
      title*=UTF-8''Du%20%C3%A4r%20ute%20p%C3%A5%20pengar.
```

Note the use of RFC5987 encoding here. Problem details SHOULD be served with Content-Language headers, even if negotiation isn't used, to aid in localisation.

The information conveyed in the detail property, if present, SHOULD focus on helping the client correct the problem, rather than giving debugging information.

Software that generates problems should note that only one problem can be sent at at time.

7. Consuming Problem Details

Consumers should use the describedby URL as the primary identifier for the problem; the title string is advisory, and included only for users who are not aware of the semantics of the URL, and don't have the ability to discover them (e.g., offline log analysis).

Likewise, the detail property SHOULD NOT be parsed for information by clients; extensions are more suitable and less error-prone ways to obtain such information.

Clients consuming problem details MUST ignore unrecognised extensions; this allows problems to evolve and include additional information in the future.

Finally, clients consuming problem details SHOULD NOT automatically deference the describedby URL; it is not intended for machine interaction.

8. Security Considerations

When defining a new problem, the information included must be carefully vetted. Likewise, when actually generating a problem - however it is serialised - the details given must also be scrutinised.

Risks include leaking information that can be exploited to compromise the system, access to the system, or the privacy of users of the system.

9. IANA Considerations

This specification defines a new Internet media type, "application/json-problem":

```
Type name: application
Subtype name: json-problem
Required parameters: None
Optional parameters: None; unrecognised parameters
                     should be ignored
Encoding considerations: Same as [RFC4627]
Security considerations: see [this document]
Interoperability considerations: None.
Published specification: [this document]
Applications that use this media type: HTTP
Additional information:
  Magic number(s): n/a
  File extension(s): n/a
  Macintosh file type code(s): n/a
Person & email address to contact for further information:
  Mark Nottingham <mnot@mnot.net>
Intended usage: COMMON
Restrictions on usage: None.
Author: Mark Nottingham <mnot@mnot.net>
Change controller: IESG
```

10. Acknowledgements

The author would like to thank Jan Algermissen, Mike Amundsen, Subbu Allamaraju, Roy Fielding, Sam Johnston, Julian Reschke, James Snell, and Erik Wilde for early review of this specification (even if some disagree with parts of it).

11. Normative References

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

[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.

[RFC5987] Reschke, J., "Character Set and Language Encoding for Hypertext Transfer Protocol (HTTP) Header Field Parameters", RFC 5987, August 2010.

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

Author's Address

Mark Nottingham
Rackspace

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