Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

A. Cedik
shipcloud GmbH
E. Wilde
Axway

November 4, 2019

Communicating Warning Information in HTTP APIs
draft-cedik-http-warning-00

Abstract

This document defines a warning code and a standard response format for warning information in HTTP APIs.

Note to Readers

This draft should be discussed on the rfc-interest mailing list (<https://www.rfc-editor.org/mailman/listinfo/rfc-interest>).

Online access to all versions and files is available on GitHub (<https://github.com/dret/I-D/tree/master/http-warning>).

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 May 7, 2020.

Table of Contents

1. Introduction

Many current APIs are based on HTTP [RFC7230] as their application protocol. Their response handling model is based on the assumption that requests either are successful or they fail. In both cases (success and fail) an HTTP status code [RFC7231] is returned to convey either fact.

But response status is not always strictly either success or failure. For example, there are cases where an underlying system returns a response with data that cannot be defined as a clear error. API providers who are integrating such a service might want to return a correct response nonetheless, but returning a HTTP status code of e.g. 200 OK without any additional information is not the only possible approach in this case.

As defined in the principles of Web architecture [W3C.REC-webarch-20041215], agents that "recover from errors by making a choice without the user's consent are not acting on the user's behalf". Therefore APIs should be able to communicate what has happened to their consumers, which then allows clients or users to make more informed decisions.

This document defines a warning code and a standard response structure for communicating and representing warning information in HTTP APIs. The goal is to allow HTTP providers to have a standardized way of communicating to their consumers that while the response can be considered to be a non-failure, there is some warning information available that they might want to take into account.

As a general guideline, warning information should be considered to be any information that can be safely ignored (treating the response as if it did not contain any warning information), but that might help clients and users to make better decisions.

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].

3. Warning Header

As described in section 5.5 of [RFC7234] the Warning header field "is used to carry additional information about the status or transformation of a message that might not be reflected in the status code". The field itself consists of the warn-code, a warn-agent, a warn-text, and an optional warn-date.

As mentioned in the introduction (Section 1), HTTP requests can be successful or they can fail. They can also result in a state where the original intent was satisfied, but a side effect happened that should be conveyed back to the client.

To make it easier for clients to handle such an event, a Warning header using the warn-code "246" and the warn-text "Embedded Warning" MAY be returned. In this case, the client MAY either treat the response according to its HTTP status code, or the client MAY use the embedded warning information to understand the nature of the warning.

The "246" warn code does not prescribe the way in which warnings are represented, but the assumption is that the response will have embedded information that allows the client to learn about the nature of the warning. The following section describes a JSON structure that MAY be used to represent the warning. HTTP services are free to use this or other formats to represent the warning information they are embedding.

4. JSON Warning Format

The JSON warning format uses the JSON format described in [RFC8259]. It is intended to be used as a building block in the response schemas of JSON-based APIs.

In many current designs of JSON-based HTTP APIs, services represent response data as members of the returned JSON object. In order to make it easier for consumers to identify information about warnings, a top-level member is defined that contains all warning information in a representation. A "warnings" member MUST encapsulate the warnings that will be returned to the client.

When an error occurred that can not be defined as a "hard error", but is meant as additional information one should consider returning this information to the APIs user. The warnings array MUST be filled with one object for each and every warning message that is returned to the client.

Entries in these individual objects follow the pattern described in [RFC7807].

When warnings are present a Warning header (as defined in Section 3) SHOULD be set to indicate that warnings have be returned. This way a client will not have to parse the response body to find out if the warnings array has entries.

5. Correlation between errors, warnings and data

5.1. Soft errors with data

Since warnings do not have an effect on the returned HTTP status code, the response status code SHOULD be in the 2xx range, indicating that the intent of the API client was successful.

```
POST /example HTTP/1.1
Host: example.com
Accept: application/json

HTTP/1.1 200 OK
Content-Type: application/json
Warning: 246 - "Embedded Warning" "Fri, 04 Oct 2019 09:59:45 GMT"

{
  "request_id": "2326b087-d64e-43bd-a557-42171155084f",
  "warnings": [
    {
      "detail": "Street name was too long. It has been shortened...",
      "instance": "https://example.com/shipments/3a186c51/msgs/c94d",
      "status": "200",
      "title": "Street name too long. It has been shortened.",
      "type": "https://example.com/errors/shortened_entry"
    },
    {
      "detail": "City for this zipcode unknown. Code for shipment..",
      "instance": "https://example.com/shipments/3a186c51/msgs/5927",
      "status": "200",
      "title": "City for zipcode unknown.",
      "type": "https://example.com/errors/city_unknown"
    }
  ],
  "id": "3a186c51d4281acb",
  "carrier_tracking_no": "84168117830018",
  "tracking_url": "http://example.com/3a186c51d",
  "label_url": "http://example.com/shipping_label_3a186c51d.pdf",
  "price": 3.4
}
```

5.2. Hard errors with warnings

As described previously, errors are exception like occurrences where processing of the request stopped and the API consumer has to be informed of this "hard error" right away.

To indicate this fact the content-type MAY be set to application/problem+json and detailed information about the error will be returned in the body following the pattern described in [RFC7807].

If warnings occurred during the processing of the request, but before the processing stopped, they SHOULD be returned alongside the errors.

```
POST /example HTTP/1.1
Host: example.com
Accept: application/json

HTTP/1.1 400 BAD REQUEST
Content-Type: application/problem+json
Warning: 246 - "Embedded Warning" "Fri, 04 Oct 2019 09:59:45 GMT"

{
  "request_id": "2326b087-d64e-43bd-a557-42171155084f",
  "detail": "The format of pickup time earliest was wrong.",
  "status": "500",
  "title": "Wrong format for pickup time",
  "type": "https://example.com/errors/wrong_format"
  "warnings": [
    {
      "detail": "Street name too long. It has been shortened to fit",
      "status": "200",
      "title": "Street name too long. It has been shortened.",
      "type": "https://example.com/errors/shortened_entry"
    }
  ]
}
```

6. Security Considerations

API providers need to exercise care when reporting warnings. Malicious actors could use this information for orchestrating attacks. Social engineering can also be a factor when warning information is returned by the API.

7. IANA Considerations

7.1. HTTP Warn Code: 246 - Embedded Warning

The HTTP warn code below has been registered by IANA per Section 7.2 of [RFC7234]:

Warn Code: 246

Short Description: Embedded Warning

Reference: Section 3 of [[this document]]

8. References

8.1. Normative References

[RFC2119] 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>.

[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, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <https://www.rfc-editor.org/info/rfc7234>.

[RFC7807] Nottingham, M. and E. Wilde, "Problem Details for HTTP APIs", RFC 7807, DOI 10.17487/RFC7807, March 2016, <https://www.rfc-editor.org/info/rfc7807>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

8.2. Informative References

[W3C.REC-webarch-20041215] Jacobs, I. and N. Walsh, "Architecture of the World Wide Web, Volume One", World Wide Web Consortium Recommendation REC-webarch-20041215, December 2004, <http://www.w3.org/TR/2004/REC-webarch-20041215>.

Appendix A. Acknowledgements

Thanks for comments and suggestions provided by ...

Authors' Addresses

Andre Cedik
shipcloud GmbH

Email: andre.cedik@googlemail.com

Erik Wilde
Axway

Email: erik.wilde@dret.net
URI: http://dret.net/netdret/