Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 1, 2020

A. Cedik
shipcloud GmbH
E. Wilde
Axway

March 30, 2020

Communicating Warning Information in HTTP APIs
draft-cedik-http-warning-01

Abstract

This document defines a new HTTP header field Content-Warning and a standard response format for representing warning information in HTTP APIs.

Note to Readers

This draft should be discussed on the rfc-interest mailing list (<https://lists.w3.org/Archives/Public/ietf-http-wg/>).

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 October 1, 2020.

Copyright Notice

Table of Contents

1. Introduction 2
2. Notational Conventions 3
3. Content-Warning Header Field 3
4. The "embedded-warning" Content-Warning Type 4
5. JSON Warning Format 4
6. Example with HTTP Header Field and Embedded Warning 5
7. Security Considerations 6
8. IANA Considerations 6
8.1. HTTP Header Field Content-Warning 6
8.2. Content-Warning Type Registry 6
8.2.1. Registration Procedure 6
8.2.2. Initial Registry Content 7
9. References 7
9.1. Normative References 7
9.2. Informative References 8
Appendix A. Acknowledgements 8
Authors' Addresses 8

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 failure) 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 success 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. Note that this specification specifically targets warnings and not errors, meaning that while it may be useful for clients to understand the warning condition and act on it, they also may choose to ignore it and treat the response as a successful one.

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 represent success, there is 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 communicate or embed 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. Content-Warning Header Field

The Content-Warning header field allows to represent different kinds of warning information via HTTP. It is defined as a Structured Header List [I-D.ietf-httpbis-header-structure]. Its ABNF is:

```
Content-Warning = sh-list
```

Each member of the list MUST have exactly the two parameters "type" and "date".

The "type" parameter represents the warning that is being signaled. Its value is defined as a sh-token and SHOULD be a type that is registered in the Content-Warning type registry Section 8.2. Clients SHOULD ignore Content-Warning types that they do not know.

The "date" parameter defines the last occurrence of this warning as a structured headers date as defined in [I-D.ietf-binary-structured-headers] (e.g. "1581410465").

4. The "embedded-warning" Content-Warning Type

This document introduces the Content-Warning Type "embedded-warning".

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, the Content-Warning type "embedded-warning" MAY be returned. In this case, the client MAY either treat the response according to its HTTP status code, or in addition the client MAY use the embedded warning information to understand the nature of the warning.

The "embedded-warning" type does not prescribe the way in which warnings are represented. 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.

An exemplary Content-Warning header field looks like this:

Content-Warning: "embedded-warning"; 1590190500

5. 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 condition occurs that can not be defined as a "hard error" (i.e., that allows clients to continue treating the resulting response as a success), additional information about this condition can be returned to the API client. The "warnings" member MUST be an array that is structured 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 the Content-Warning header field (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 whether a warnings member is present.

6. Example with HTTP Header Field and Embedded Warning

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
Content-Warning: "embedded-warning"; 1590190500

{
  "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
}
```

7. 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.

8. IANA Considerations

8.1. HTTP Header Field Content-Warning

This specification registers the following entry in the Permanent Message Header Field Names registry established by [RFC3864]:

- Header field name: Content-Warning
- Applicable protocol: HTTP
- Status: standard
- Author/Change Controller: IETF
- Specification document(s): [this document]
- Related information:

8.2. Content-Warning Type Registry

The "Content-Warning Type Registry" defines the namespace for new Content-Warning types. This specification establishes a new registry according to the guidelines given in [RFC8126]. This new registry should not be included in an existing group of registries.

8.2.1. Registration Procedure

A registration MUST include the following fields:

- Content-Warning Type: Name of the Content-Warning Type
- Reference: Pointer to a specification text

The registration policy for this registry is "Specification Required" as defined by [RFC8126], Section 4.6. They MUST follow the "sh-token" syntax defined by [I-D.ietf-httpbis-header-structure].

8.2.2. Initial Registry Content

The registry has been populated with the registered values shown below:

Content-Warning Type	Reference
embedded-warning	this RFC, Section 4

9. References

9.1. Normative References

[I-D.ietf-binary-structured-headers]
Nottingham, M., "Binary Structured HTTP Headers", draft-nottingham-binary-structured-headers-02 (work in progress), March 2020.

[I-D.ietf-httpbis-header-structure]
Nottingham, M. and P. Kamp, "Structured Headers for HTTP", draft-ietf-httpbis-header-structure-14 (work in progress), October 2019.

[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>.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <https://www.rfc-editor.org/info/rfc3864>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

9.2. 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/