Workgroup: Network Working Group
Internet-Draft:
draft-divilly-user-defined-resource-error-00
Published: 26 March 2020
Intended Status: Informational
Expires: 27 September 2020
Authors: C. Divilly
Oracle

User Defined Resource Error HTTP Status Code

Abstract

This document specifies an additional HyperText Transfer Protocol (HTTP) status code to indicate server error conditions arising during evaluation of a user defined resource hosted by the server, rather than in the server itself.

Conventions and Terminology

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 RFC 2119 [RFC2119].

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 27 September 2020.

Copyright Notice

Table of Contents

1. Note to Readers

Per [RFC7231] Section 8.2.2 this document avoids allocating a specific number for the proposed new HTTP status code until there is clear consensus that it will be registered. The code 5NN is used throughout this document to denote this new status code.

2. Introduction

Some HTTP servers offer mechanisms for users to define their own programmatically generated resources. This specification terms such a resource as a 'User Defined Resource'. In such cases it is useful to distinguish between errors arising due to defects in the User Defined Resource and errors arising due to defects in the server itself.

This document proposes a new 5NN HTTP status code. This status code indicates that an error occurred when the server attempted to produce a representation of the User Defined Resource, and the error occurred when attempting to evaluate the program that generates the resource, rather than an error condition in the server itself.

2.1. Why does 500 Internal Server Error not suffice?

This section is non normative.

The current state of the art is to represent errors in User Defined Resources as a 500 Internal Server Error status. In the author's experience this is not optimal for the following reasons:

- It is widely understood that a 500 Internal Server Error represents a serious error condition that likely needs remediation by the server's operators

- Error conditions in User Defined Resources are frequent and expected. In a well architected system with isolation between the environment executing the User Defined Resource program and the server hosting the User Defined Resource, errors should be benign and not require any remediation by the server's operators

In the author's own experience we have attempted to address this by taking two actions:

1. Add an additional response header to enable tools to detect that the error condition relates to a user defined resource and should not be treated as an error that requires remediation

2. Add explanatory text to the response body to communicate to the end user that the error represents a problem in a User Defined Resource

Our experience is that these approaches have very limited effectiveness:

- The additional response header is lost in access logs which are often the resource that is used for monitoring the status of the server.

- Users do not read or understand the explanatory text well enough. They see the well known 500 Internal Server Error status code and feel they understand that this means there is a problem with the server. Too often they proceed to filing support tickets against the server operator, rather than against the developer responsible for the hosted User Defined Resource. This wastes the user's, server operators', and User Defined Resource developer's time and resources.

We believe there is substantial value in assigning a new 5NN class HTTP status code for this class of server error. It will be a very clear signal to both tools and users that the error condition needs to be handled in a distinctly different manner to how 500 Internal Server Error conditions are handled.

Traditionally there has been some kind of direct relationship between the author of server resources and the operator of the server. With the rise of multi-tenant hosted platforms (such as 'serverless' plaforms) increasingly there is no direct relationship between the party hosting the User Defined Resource and the party that authored the User Defined Resource, and thus it becomes appropriate to distinguish at the HTTP status code level between these two classes of error.

3. 5NN User Defined Resource Error

The 5NN (User Defined Resource Error) status code indicates that the server encountered an unexpected condition when evaluating a User Defined Resource that prevented the server from fulfilling the request.

A 5NN response is not cacheable.

The response message **MAY** contain information that identifies the User Defined Resource that originated the error. The response message **SHOULD** contain additional information that can help the author of User Defined Resource diagnose the root cause of the error.

The response **SHOULD** include an identifier that uniquely identifies the error condition instance. This identifier should also appear with any log messages or other diagnostic information that the server produces.

The response **MAY** include a URI [RFC3986] that points to a resource that the User Defined Resource author can use to review the log and other diagnostic information associated with the error condition. Access to this URI **MUST** be restricted to ensure only the User Defined Resource author can access it.

It is **RECOMMENDED** that the server provide the User Defined Resource author with secured access to the logs pertaining to the error instance, and a capability to filter/search these logs keyed by the error identifier.

The log information **SHOULD** provide detailed information about the nature and origin of the error, to enable the User Defined Resource author to diagnose the root cause of the error, whereas the error response **SHOULD** contain the minimal information required to identify the corresponding log messages.

3.1. Relationship to 500 Internal Server Error

The 5NN status code can be considered a specialization of the 500 status code. To quote the HTTP Specification [RFC7231]:

> HTTP status codes are extensible. HTTP clients are not required to understand the meaning of all registered status codes, though such understanding is obviously desirable. However, a client MUST understand the class of any status code, as indicated by the first digit, and treat an unrecognized status code as being equivalent to the x00 status code of that class

Thus clients **SHOULD** treat the 5NN status code in the same manner as they treat the 500 status code.

The primary value of the 5NN status code is to enable efficient routing of problem reports to the party best placed to remediate the error condition.

In the case of a 5NN status code the party best placed to remediate the error condition is the author of the User Defined Resource.

In the case of a 500 status code the party best placed to remediate the error condition is the server operator.

A 500 status is unexpected and likely requires a corrective action from the server operators, as the error may indicate a threat to the stability and availability of the server.

A 5NN status is likely to be commonplace, as User Defined Resource authors will be expected to make mistakes when authoring those resources. Assuming a well architected server with proper isolation between the server and the User Defined Resources, such error conditions are unlikely to be a threat to the stability and availability of the server.

The ability to distinguish between 500 and 5NN status codes provides a strong signal enabling both tools and humans to respond to the appropriate party to remediate the error condition.

4. IANA Considerations

The HTTP Status Codes Registry should be updated with the following entry:

*Code: 5NN

*Description: User Defined Resource Error

*Specification: [this document]

5. Security Considerations

When the server includes information that identifies the User Defined Resource that caused the error, or additional information

that helps the author diagnose the root cause, care must be taken not to disclose information that may be useful to an attacker.

Care needs to be taken to ensure that the log messages do not reveal sensitive information about the users of the User Defined Resource, see [RFC7230] section 9.8 for relevant guidance on this topic.

6. Example

This section is non-normative.

Below is an example response that leverages the Problem Details for HTTP APIs syntax [RFC7807] to communicate information about the error condition:

```
HTTP/1.1 5NN User Defined Resource Error
Content-Type: application/problem+json
Content-Language: en

{
 "type":     "https://example.com/errs/user-defined-resource-error",
 "title":    "User Defined Resource Error",
 "detail":   "An unexpected error condition occurred when
              evaluating a user defined resource",
 "trace_id": "a75382c4-d61d-4c16-8dde-a01afc7b51a2",
 "instance": "/logs/?trace_id=a75382c4-d61d-4c16-8dde-a01afc7b51a2"
}
```

*The detail message is careful to reveal minimal information about the User Defined Resource that experienced the error condition.

*The trace_id field provides a unique identifier for the error condition that can be used to correlate corresponding log entries and other diagnostic information relevant to this error condition.

*The instance URI points to a (secured) resource that can be interrogated to view all the log messages associated with this specific error instance.

7. Normative References

[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>.

[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>.

[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>.

[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, <https://www.rfc-editor.org/info/rfc3986>.

[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>.

Author's Address

Colm Divilly
Oracle

Email: colm.divilly@oracle.com