Building Blocks for HTTP APIs
Internet-Draft
Intended status: Standards Track
Expires: 13 August 2026

M. Kleidl
Transloadit
L. Pardue
Cloudflare
R. Polli
Par-Tec

9 February 2026

HTTP Problem Types for Digest Fields
draft-ietf-httpapi-digest-fields-problem-types-04

Abstract

This document specifies HTTP problem types that servers can use in responses to problems encountered while dealing with a request carrying integrity fields and integrity preference fields. Using an HTTP problem type, the server can provide machine-readable error details to aid debugging or error reporting.

About This Document

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

The latest revision of this draft can be found at https://ietf-wg-httpapi.github.io/digest-fields-problem-types/draft-ietf-httpapi-digest-fields-problem-types.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-httpapi-digest-fields-problem-types/.

Discussion of this document takes place on the Building Blocks for HTTP APIs Working Group mailing list (mailto:httpapi@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/httpapi/. Subscribe at https://www.ietf.org/mailman/listinfo/httpapi/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-httpapi/digest-fields-problem-types.

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/.

Copyright Notice

Table of Contents

1. Introduction

[DIGEST] and [UNENC-DIGEST] define HTTP fields for exchanging integrity digests and preferences, but do not specify, require, or recommend any specific behavior for error handling relating to integrity by design. The responsibility is instead delegated to applications. This document defines a set of HTTP problem types ([PROBLEM]) that can be used by server applications to indicate that a problem was encountered while dealing with a request carrying

integrity fields and integrity preference fields.

For example, a request message may include content alongside Content-Digest and Repr-Digest fields that use a digest algorithm the server does not support. An application could decide to reject this request because it cannot validate the integrity. Using an HTTP problem type, the server can provide machine-readable error details to aid debugging or error reporting, as shown in the following example.

```
# NOTE: '\' line wrapping per RFC 8792

HTTP/1.1 400 Bad Request
Content-Type: application/problem+json
Want-Content-Digest: sha-512=3, sha-256=10

{
  "type": "https://iana.org/assignments/http-problem-types#\
    digest-unsupported-algorithms",
  "title": "Unsupported hashing algorithms",
  "unsupported_algorithms": [
    {
      "algorithm": "foo",
      "header": "Want-Content-Digest"
    }
  ]
}
```

2. Conventions and Definitions

Some examples in this document contain long lines that may be folded, as described in [RFC8792].

The terms "integrity fields" and "integrity preference fields" in this document are to be interpreted as described in [DIGEST] and updated in [UNENC-DIGEST].

The term "problem type" in this document is to be interpreted as described in [PROBLEM].

The terms "request", "response", "intermediary", "sender", "client", and "server" are from [HTTP].

The problem types in this document are defined using JSON [JSON]. They can be serialized into an equivalent XML format as outlined in Appendix B of [PROBLEM].

3. Problem Types

The following section defines three problem types to express common problems that occur when handling integrity or integrity preference fields on the server. These problem types use the digest- prefix in their type URI. Other problem types that are defined outside this document, yet specific to digest related problems, may reuse this prefix.

Requests can include multiple integrity or integrity preference fields. For example, they may use the Content-Digest and Repr-Digest fields simultaneously or express preferences for content and representation digests at the same time. Therefore, similar problems can appear multiple times for one request. The problem types defined in this document allow expressing multiple appearances, while each time identifying the corresponding header that contained the problematic value.

3.1. Unsupported Hashing Algorithms

This section defines the "https://iana.org/assignments/http-problem-types#digest-unsupported-algorithms" problem type. A server can use this problem type to communicate to the client that one or more of the hashing algorithms referenced in the integrity or integrity preference fields present in the request are not supported.

For this problem type, the unsupported_algorithms extension member is defined, whose value is a JSON [JSON] array of entries identifying each unsupported algorithm. Each entry in the array is a JSON object with the following members:

* The algorithm member is a JSON string containing the algorithm key of the unsupported algorithm.

* The header member is a JSON string containing the name of the integrity or integrity preference field that referenced the unsupported algorithm.

The response can include the corresponding integrity preference field to indicate the server’s algorithm support and preference.

This problem type is a hint to the client about algorithm support, which the client could use to retry the request with different, supported algorithms.

The following example shows a request with the content {"title": "New Title"} (plus LF). The digest fields use the MD5 algorithm, which is not supported by the server as the algorithm is deprecated.

```
POST /books HTTP/1.1
Host: foo.example
Content-Type: application/json
Accept: application/json
Repr-Digest: md5=:Uwq9xB4MJtDTknVOSEE1WA==:
Content-Digest: md5=:Uwq9xB4MJtDTknVOSEE1WA==:
Unencoded-Digest: md5=:Uwq9xB4MJtDTknVOSEE1WA==:

{"title": "New Title"}
```

Figure 1: A request with md5 integrity fields, which are not supported by the server

```
# NOTE: '\' line wrapping per RFC 8792

HTTP/1.1 400 Bad Request
Content-Type: application/problem+json
Want-Repr-Digest: sha-512=10, md5=0
Want-Content-Digest: sha-512=10, md5=0
Want-Unencoded-Digest: sha-512=10, md5=0

{
  "type": "https://iana.org/assignments/http-problem-types#\
    digest-unsupported-algorithms",
  "title": "Unsupported hashing algorithms",
  "unsupported_algorithms": [
    {
      "algorithm": "md5",
      "header": "Repr-Digest"
    },
    {
      "algorithm": "md5",
      "header": "Content-Digest"
    },
    {
      "algorithm": "md5",
      "header": "Unencoded-Digest"
    }
  ]
}
```

Figure 2: Response indicating the problem and advertising the supported algorithms

This problem type can also be used when a request contains an integrity preference field with an unsupported algorithm. For example:

```
GET /items/123 HTTP/1.1
Host: foo.example
Want-Repr-Digest: md5=10
```

Figure 3: A request with a md5 integrity preference field, which is not supported by the server

```
# NOTE: '\' line wrapping per RFC 8792

HTTP/1.1 400 Bad Request
Content-Type: application/problem+json

{
  "type": "https://iana.org/assignments/http-problem-types#\
    digest-unsupported-algorithms",
  "title": "Unsupported hashing algorithms",
  "unsupported_algorithms": [
    {
      "algorithm": "md5",
      "header": "Want-Repr-Digest"
    }
  ]
}
```

Figure 4: Response indicating the problem

3.2. Invalid Digest Values

This section defines the "https://iana.org/assignments/http-problem-types#digest-invalid-values" problem type. A server can use this problem type when responding to a request, whose integrity fields include a digest value that cannot be generated by the corresponding hashing algorithm. For example, if the digest value of the sha-512 hashing algorithm is not 64 bytes long, it cannot be a valid SHA-512 digest value and the server can skip computing the digest value. This problem type cannot be used if the server is not able to parse the integrity fields and obtain a value according to Section 4.5 of [STRUCTURED-FIELDS], for example because of a syntax error.

For this problem type, the invalid_digests extension member is defined, whose value is a JSON [JSON] array of entries identifying each invalid digest. Each entry in the array is a JSON object with the following members:

* The algorithm member is a JSON string containing the algorithm key.

* The header member is a JSON string containing the name of the integrity field that contained the invalid digest value.

* The reason member is a JSON string containing a human-readable description why the value is considered invalid.

This problem type indicates a fault in the sender's calculation or encoding of the digest value. A retry of the same request without modification will likely not yield a successful response.

The following example shows a request with the content {"hello": "world"} (plus LF), but the digest has been truncated. The subsequent response indicates the invalid SHA-512 digest.

```
PUT /items/123 HTTP/1.1
Host: foo.example
Content-Type: application/json
Repr-Digest: sha-512=:YMAam51Jz/jOATT6/zvHrLVgOYTGFy1d6GJiOHTohq4:

{"hello": "world"}
```

Figure 5: A request with a sha-512 integrity field, whose digest has been truncated to 32 bytes

```
# NOTE: '\' line wrapping per RFC 8792

HTTP/1.1 400 Bad Request
Content-Type: application/problem+json

{
  "type": "https://iana.org/assignments/http-problem-types#\
    digest-invalid-values",
  "title": "Invalid digest values",
  "invalid_digests": [
    {
      "algorithm": "sha-512",
      "header": "Repr-Digest",
      "reason": "digest value is not 64 bytes long"
    }
  ]
}
```

Figure 6: Response indicating that the provided digest is too short

3.3. Mismatched Digest Values

This section defines the "https://iana.org/assignments/http-problem-types#digest-mismatched-values" problem type. A server can use this problem type when responding to a request, whose integrity fields include a digest value that does not match the digest value that the server calculated for the request content or representation.

For this problem type, the mismatched_digests extension member is defined, whose value is a JSON [JSON] array of entries identifying each mismatched digest. Each entry in the array is a JSON object with the following members:

* The algorithm member is a JSON string containing the algorithm key of the hashing algorithm.

* The provided_digest member is a JSON string containing the digest value taken from the request's integrity fields. The digest value is serialized as a byte sequence as described in Section 4.1.8 of [STRUCTURED-FIELDS].

* The header member is a JSON string containing the name of the integrity field that contained the mismatched digest value.

The problem type intentionally does not include the digest value calculated by the server to avoid attackers abusing this information for oracle attacks.

If the sender receives this problem type, the request might be modified unintentionally by an intermediary. The sender could use this information to retry the request without modification to address temporary transmission issues.

The following example shows a request with the content {"hello": "woXYZ"} (plus LF), but the representation digest for {"hello": "world"} (plus LF). The subsequent response indicates the mismatched SHA-256 digest value.

```
PUT /items/123 HTTP/1.1
Host: foo.example
Content-Type: application/json
Repr-Digest: sha-256=:RK/0qy18MlBSVnWgjwz6lZEWjP/lF5HF9bvEF8FabDg=:

{"hello": "woXYZ"}
```

Figure 7: A request with a sha-256 integrity field, which does not belong to the representation

```
# NOTE: '\' line wrapping per RFC 8792

HTTP/1.1 400 Bad Request
Content-Type: application/problem+json

{
  "type": "https://iana.org/assignments/http-problem-types#\
    digest-mismatched-values",
  "title": "Mismatched digest values",
  "mismatched_digests": [
    {
      "algorithm": "sha-256",
      "provided_digest": \
        ":RK/0qy18MlBSVnWgjwz6lZEWjP/lF5HF9bvEF8FabDg=:",
      "header": "Repr-Digest"
    }
  ]
}
```

Figure 8: Response indicating the mismatched digests

4. Security Considerations

All security considerations from Section 6 of [DIGEST] and Section 7 of [UNENC-DIGEST] apply as well.

Disclosing error details could leak information such as the presence of intermediaries or the server's implementation details. Moreover, they can be used to fingerprint the server.

To mitigate these risks, server operators could assess the risk of disclosing error details and prefer a general problem type over a more specific one.

There is no method defined for the server to communicate a digest value that it calculated for the purpose of validation. Such information can be abused for oracle attacks.

5. IANA Considerations

IANA is asked to register the following entries in the "HTTP Problem Types" registry at https://www.iana.org/assignments/http-problem-types (https://www.iana.org/assignments/http-problem-types).

5.1. Registration of "digest-unsupported-algorithms" Problem Type

Type URI: https://iana.org/assignments/http-problem-types#digest-unsupported-algorithms

Title: Unsupported Hashing Algorithms

Recommended HTTP status code: 400

Reference: Section 3.1 of this document

5.2. Registration of "digest-invalid-values" Problem Type

Type URI: https://iana.org/assignments/http-problem-types#digest-invalid-values

Title: Invalid Digest Values

Recommended HTTP status code: 400

Reference: Section 3.2 of this document

5.3. Registration of "digest-mismatched-values" Problem Type

Type URI: https://iana.org/assignments/http-problem-types#digest-mismatched-values

Title: Mismatched Digest Values

Recommended HTTP status code: 400

Reference: Section 3.3 of this document

6. Normative References

[DIGEST] Polli, R. and L. Pardue, "Digest Fields", RFC 9530, DOI 10.17487/RFC9530, February 2024, <https://www.rfc-editor.org/rfc/rfc9530>.

[HTTP] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[JSON] 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/rfc/rfc8259>.

[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>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/rfc/rfc8792>.

[STRUCTURED-FIELDS] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", RFC 9651, DOI 10.17487/RFC9651, September 2024, <https://www.rfc-editor.org/rfc/rfc9651>.

[UNENC-DIGEST] Pardue, L. and M. West, "HTTP Unencoded Digest", Work in Progress, Internet-Draft, draft-ietf-httpbis-unencoded-digest-03, 12 December 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-unencoded-digest-03>.

Authors' Addresses

Marius Kleidl
Transloadit
Email: marius@transloadit.com

Lucas Pardue
Cloudflare
Email: lucas@lucaspardue.com

Roberto Polli
Par-Tec
Italy
Email: robipolli@gmail.com