

CoRE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 7 November 2022

T. Fossati
arm
C. Bormann
Universität Bremen TZI
6 May 2022

Concise Problem Details For CoAP APIs
draft-ietf-core-problem-details-03

Abstract

This document defines a "problem detail" as a way to carry machine-readable details of errors in a REST response to avoid the need to define new error response formats for REST APIs. The format is inspired by, but intended to be more concise than, the Problem Details for HTTP APIs defined in [RFC 7807](#).

Discussion Venues

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

Discussion of this document takes place on the CORE Working Group mailing list (core@ietf.org), which is archived at <https://mailarchive.ietf.org/arch/browse/core/>.

Source for this draft and an issue tracker can be found at <https://github.com/core-wg/core-problem-details>.

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 7 November 2022.

Internet-Draft

CoRE Problem Details

May 2022

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Revised BSD License.

Table of Contents

1.	Introduction	3
1.1.	Requirements Language	3
2.	Basic Problem Details	3
3.	Extending Concise Problem Details	5
3.1.	Standard Problem Detail Entries	5
3.2.	Custom Problem Detail Entries	6
4.	Security Considerations	8
5.	IANA Considerations	8
5.1.	Standard Problem Detail Key registry	9
5.2.	Custom Problem Detail Key registry	10
5.3.	Media Type	11
5.4.	Content-Format	12
5.5.	CBOR Tag 38	12
6.	References	12
6.1.	Normative References	12
6.2.	Informative References	13
Appendix A.	Language-Tagged Strings	14
A.1.	Introduction	14
A.2.	Detailed Semantics	14
A.3.	Examples	17
Appendix B.	Interworking with RFC 7807	18
	Acknowledgments	18
	Contributors	19
	Authors' Addresses	19

1. Introduction

REST response status information such as CoAP response codes ([Section 5.9 of \[RFC7252\]](#)) is sometimes not sufficient to convey enough information about an error to be helpful. This specification defines a simple and extensible framework to define CBOR [\[STD94\]](#) data items to suit this purpose. It is designed to be reused by REST APIs, which can identify distinct "problem types" specific to their needs. Thus, API clients can be informed of both the high-level error class (using the response code) and the finer-grained details of the problem (using this vocabulary), as shown in Figure 1.

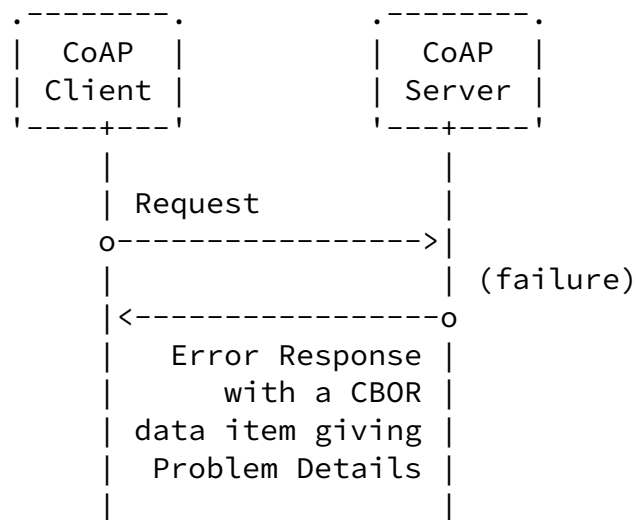


Figure 1: Problem Details: Example with CoAP

The framework presented is largely inspired by the Problem Details for HTTP APIs defined in [\[RFC7807\]](#). [Appendix B](#) discusses applications where interworking with [\[RFC7807\]](#) is required.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",

"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14](#) [[RFC2119](#)] [[RFC8174](#)] when, and only when, they appear in all capitals, as shown here.

2. Basic Problem Details

A Concise Problem Details data item is a CBOR data item with the following structure (notated in CDDL [[RFC8610](#)]):

Fossati & Bormann	Expires 7 November 2022	[Page 3]
-------------------	-------------------------	----------

Internet-Draft	CoRE Problem Details	May 2022
----------------	----------------------	----------

```
problem-details = non-empty<{
  ? &(title: -1) => oltext
  ? &(detail: -2) => oltext
  ? &(instance: -3) => ~uri
  ? &(response-code: -4) => uint .size 1
  standard-problem-detail-entries
  custom-problem-detail-entries
}>

standard-problem-detail-entries = (
  * nint => any
)

custom-problem-detail-entries = (
  * (uint/~uri) => { + any => any }
)

non-empty<M> = (M) .and ({ + any => any })

oltext = text / tag38 ; see Appendix A for tag38
```

Figure 2: Problem Detail Data Item

A number of problem detail entries, the Standard Problem Detail entries, are predefined (more predefined details can be registered, see [Section 3.1](#)):

The title (key -1):

A short, human-readable summary of the problem type. It SHOULD

NOT change from occurrence to occurrence of the problem.

The detail (key -2):

A human-readable explanation specific to this occurrence of the problem.

The instance (key -3):

A URI reference that identifies the specific occurrence of the problem. It may or may not yield further information if dereferenced.

The response-code (key -4)

The CoAP response code ([Section 5.9 of \[RFC7252\]](#)) generated by the origin server for this occurrence of the problem.

Note that, unlike [\[RFC7807\]](#), Concise Problem Details data items have no explicit type.

Both "title" and "detail" can use either an unadorned CBOR text string (text) or a language-tagged text string (tag38); see [Appendix A](#) for the definition of the latter.

The "title" string is advisory and included to give consumers a shorthand for the category of the error encountered.

The "detail" member, if present, ought to focus on helping the client correct the problem, rather than giving debugging information. Consumers SHOULD NOT parse the "detail" member for information; extensions (see [Section 3](#)) are more suitable and less error-prone ways to obtain such information.

Note that the "instance" URI reference may be relative; this means that it must be resolved relative to the representation's base URI, as per [Section 5 of \[STD66\]](#).

Note that the "response code" value is a numeric representation of the actual code, so it does not take the usual form that resembles an HTTP status code -- 4.04 Not found is represented by the number 132.

[3.](#) Extending Concise Problem Details

This specification defines a generic problem type container with only a minimal set of attributes to make it usable.

It is expected that applications will extend the base format by defining new attributes.

These new attributes fall into two categories: generic and application specific.

Generic attributes will be allocated in the standard-problem-detail-entries slot according to the registration procedure defined in [Section 3.1](#).

Application-specific attributes will be allocated in the custom-problem-detail-entries slot according to the procedure described in [Section 3.2](#).

[3.1](#). Standard Problem Detail Entries

Beyond the Standard Problem Detail keys defined in Figure 2, additional Standard Problem Detail keys can be registered for use in the standard-problem-detail-entries slot (see [Section 5.1](#)).

Standard Problem Detail keys are negative integers, so they never can conflict with Custom Problem Detail keys defined for a specific application domain (which are unsigned integers or URIs.)

In summary, the keys for Standard Problem Detail entries are in a global namespace that is not specific to a particular application domain.

Consumers of a Concise Problem Details instance MUST ignore any Standard Problem Detail entries that they do not recognize; this allows problem details to evolve.

[3.2](#). Custom Problem Detail Entries

Applications may extend the Problem Details data item with additional

entries to convey additional, application-specific information.

Such new entries are allocated in the custom-problem-detail-entries slot, and carry a nested map specific to that application. The map key can either be an (absolute!) URI (controlled by the entity defining this extension), or an unsigned integer. Only the latter needs to be registered ([Section 5.2](#)).

Within the nested map any number of attributes can be given for a single extension. The semantics of each custom attribute MUST be described in the documentation for the extension; for extension that are registered (i.e., are identified by an unsigned int) that documentation goes along with the registration.

The unsigned integer form allows a more compact representation, in exchange, authors are expected to comply with the required registration and documentation process. In comparison, the URI form is less space-efficient but requires no registration. It is therefore useful for experimenting during the development cycle and for applications deployed in environments where producers and consumers of Concise Problem Details are more tightly integrated. (The URI form thus covers the potential need we might otherwise have for a "private use" range for the unsigned integers.)

Note that the URI given for the extension is for identification purposes only and, even if dereferenceable in principle, MUST NOT be dereferenced in the normal course of handling problem details (i.e., outside diagnostic/debugging procedures involving humans).

An example of a custom extension using a URI as custom-problem-detail-entries key is shown in Figure 3.

```
{
  / title /           -1: "title of the error",
  / detail /          -2: "detailed information about the error",
  / instance /        -3: "coaps://pd.example/FA317434",
  / response-code /   -4: 128, / 4.00 /

  "tag:3gpp.org,2022-03:TS29112": {
    / cause /    0: "machine readable error cause",
```

```

    / invalidParams / 1: [
      [
        / param / "first parameter name",
        / reason / "must be a positive integer"
      ],
      [
        / param / "second parameter name"
      ]
    ],
    / supportedFeatures / 2: "d34db33f"
  }
}

```

Figure 3: Example Extension with URI key

Obviously, an SDO like 3GPP can also easily register such a custom problem detail entry to receive a more efficient unsigned integer key; the same example but using a registered unsigned int as custom-problem-detail-entries key is shown in Figure 4.


```

/ title /          -1: "title of the error",
/ detail /         -2: "detailed information about the error",
/ instance /       -3: "coaps://pd.example/FA317434",
/ response-code / -4: 128, / 4.00 /

/example value 4711 not actually registered like this:/
4711: {
  / cause / 0: "machine readable error cause",
  / invalidParams / 1: [
    [
      / param / "first parameter name",
      / reason / "must be a positive integer"
    ],
    [
      / param / "second parameter name"
    ]
  ],
  / supportedFeatures / 2: "d34db33f"
}
}

```

Figure 4: Example Extension with unsigned int (registered) key

In summary, the keys for the maps used inside Custom Problem Detail entries are defined specifically to the identifier of that Custom Problem Detail entry, the documentation of which defines these internal entries, typically chosen to address a given application domain. Consumers of a Concise Problem Details instance **MUST** ignore any Custom Problem Detail entries, or keys inside the Custom Problem Detail entries, that they do not recognize; this allows Custom Problem Detail entries to evolve and include additional information in the future.

[4.](#) Security Considerations

The security and privacy considerations outlined in [Section 5 of \[RFC7807\]](#) apply in full.

[5.](#) IANA Considerations

// RFC Editor: please replace RFC XXXX with this RFC number and
 // remove this note.

[5.1.](#) Standard Problem Detail Key registry

This specification defines a new sub-registry for Standard Problem Detail Keys in the CoRE Parameters registry [[IANA.core-parameters](#)], with the policy "specification required" [[RFC8126](#)].

Each entry in the registry must include:

Key value:

a negative integer to be used as the value of the key

Name:

a name that could be used in implementations for the key

CDDL type:

type of the data associated with the key in CDDL notation

Brief description:

a brief description

reference:

a reference document

Initial entries in this sub-registry are as follows:

Internet-Draft

CoRE Problem Details

May 2022

Key value	Name	CDDL Type	Brief description	Reference
-1	title	text	short, human-readable summary of the problem type	RFCXXXX
-2	detail	text	human-readable explanation specific to this occurrence of the problem	RFCXXXX
-3	instance	~uri	URI reference identifying specific occurrence of the problem	RFCXXXX
-4	response-code	uint .size 1	CoAP response code	RFCXXXX

Table 1: Initial Entries in Standard Problem Detail Key registry

5.2. Custom Problem Detail Key registry

This specification defines a new sub-registry for Custom Problem Detail Keys in the CoRE Parameters registry [[IANA.core-parameters](#)], with the policy "first come first served" [[RFC8126](#)].

Each entry in the registry must include:

Key value:

an unsigned integer to be used as the value of the key

Name:

a name that could be used in implementations for the key

Brief description:
a brief description

Reference:
a reference document that provides a description of the map,
including a CDDL description, that describes all inside keys and
values

Initial entries in this sub-registry are as follows:

Fossati & Bormann Expires 7 November 2022 [Page 10]

Internet-Draft CoRE Problem Details May 2022

Key value	Name	Brief description	Reference
7807	tunnel-7807	Carry RFC 7807 problem details in a Concise Problem Details data item	RFCXXXX

Table 2: Initial Entries in Custom Problem Detail Key registry

5.3. Media Type

IANA is requested to add the following Media-Type to the "Media
Types" registry [[IANA.media-types](#)].

Name	Template	Reference
concise-problem-details+cbor	application/concise-problem- details+cbor	RFCXXXX, Section 5.3

Table 3: New Media Type application/concise-problem-details+cbor

Type name: application
Subtype name: concise-problem-details+cbor
Required parameters: none
Optional parameters: none
Encoding considerations: binary (CBOR data item)
Security considerations: [Section 4](#) of RFC XXXX

Interoperability considerations: none
 Published specification: [Section 5.3](#) of RFC XXXX
 Applications that use this media type: Clients and servers in the Internet of Things
 Fragment identifier considerations: The syntax and semantics of fragment identifiers is as specified for "application/cbor". (At publication of RFC XXXX, there is no fragment identification syntax defined for "application/cbor".)
 Person & email address to contact for further information: CoRE WG mailing list (core@ietf.org), or IETF Applications and Real-Time Area (art@ietf.org)
 Intended usage: COMMON
 Restrictions on usage: none
 Author/Change controller: IETF
 Provisional registration: no

Fossati & Bormann Expires 7 November 2022 [Page 11]

Internet-Draft CoRE Problem Details May 2022

[5.4.](#) Content-Format

IANA is requested to register a Content-Format number in the "CoAP Content-Formats" sub-registry, within the "Constrained RESTful Environments (CoRE) Parameters" Registry [[IANA.core-parameters](#)], as follows:

Content-Type	Content Coding	ID	Reference
application/concise-problem- details+cbor	-	TBD1	RFC XXXX

Table 4: New Content-Format

TBD1 is to be assigned from the space 256..999.

In the registry as defined by [Section 12.3 of \[RFC7252\]](#) at the time of writing, the column "Content-Type" is called "Media type" and the column "Content Coding" is called "Encoding".

// This paragraph to be removed by RFC editor.

[5.5.](#) CBOR Tag 38

In the registry "CBOR Tags" [[IANA.cbor-tags](#)], IANA has registered CBOR Tag 38. IANA is requested to replace the reference for this registration with [Appendix A](#), RFC XXXX.

6. References

6.1. Normative References

- [IANA.cbor-tags]
IANA, "Concise Binary Object Representation (CBOR) Tags",
<<https://www.iana.org/assignments/cbor-tags>>.
- [IANA.core-parameters]
IANA, "Constrained RESTful Environments (CoRE)
Parameters",
<<https://www.iana.org/assignments/core-parameters>>.
- [IANA.media-types]
IANA, "Media Types",
<<https://www.iana.org/assignments/media-types>>.

- [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/rfc/rfc2119>>.
- [RFC4647] Phillips, A., Ed. and M. Davis, Ed., "Matching of Language Tags", [BCP 47](#), [RFC 4647](#), DOI 10.17487/RFC4647, September 2006, <<https://www.rfc-editor.org/rfc/rfc4647>>.
- [RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", [BCP 47](#), [RFC 5646](#), DOI 10.17487/RFC5646, September 2009, <<https://www.rfc-editor.org/rfc/rfc5646>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", [RFC 7252](#), DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/rfc/rfc7252>>.

- [RFC7807] Nottingham, M. and E. Wilde, "Problem Details for HTTP APIs", [RFC 7807](#), DOI 10.17487/RFC7807, March 2016, <<https://www.rfc-editor.org/rfc/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/rfc/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", [RFC 8610](#), DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/rfc/rfc8610>>.
- [STD66] 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/rfc/rfc3986>>.
- [STD94] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, [RFC 8949](#), DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/rfc/rfc8949>>.

6.2. Informative References

Fossati & Bormann Expires 7 November 2022 [Page 13]

Internet-Draft CoRE Problem Details May 2022

- [I-D.ietf-httpapi-rfc7807bis] Nottingham, M., Wilde, E., and S. Dalal, "Problem Details for HTTP APIs", Work in Progress, Internet-Draft, [draft-ietf-httpapi-rfc7807bis-02](#), 16 April 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-httpapi-rfc7807bis-02>>.
- [RDF] Cyganiak, R., Wood, D., and M. Lanthaler, "RDF 1.1 Concepts and Abstract Syntax", W3C Recommendation, 25 February 2014, <<http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", [RFC 4648](#), DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/rfc/rfc4648>>.

[Unicode-14.0.0]

The Unicode Consortium, "The Unicode Standard, Version 14.0.0", Mountain View: The Unicode Consortium, ISBN 978-1-936213-29-0, September 2021, <<https://www.unicode.org/versions/Unicode14.0.0/>>. Note that while this document references a version that was recent at the time of writing, the statements made based on this version are expected to remain valid for future versions.

[Appendix A](#). Language-Tagged Strings

This appendix serves as the archival documentation for CBOR Tag 38, a tag for serializing language-tagged text strings in CBOR. The text of this appendix is adapted from the specification text supplied for its initial registration. It has been extended to allow supplementing the language tag by a direction indication.

[A.1](#). Introduction

In some cases it is useful to specify the natural language of a text string. This specification defines a tag that does just that. One technology that supports language-tagged strings is the Resource Description Framework (RDF) [[RDF](#)].

[A.2](#). Detailed Semantics

A language-tagged string in CBOR has the tag 38 and consists of an array with a length of 2 or 3.

The first element is a well-formed language tag under Best Current Practice 47 ([[RFC5646](#)] and [[RFC4647](#)]), represented as a UTF-8 text string (major type 3).

The second element is an arbitrary UTF-8 text string (major type 3). Both the language tag and the arbitrary string can optionally be annotated with CBOR tags; this is not shown in the CDDL below.

The optional third element, if present, is a Boolean value that indicates a direction: false for "ltr" direction, true for "rtl" direction. If the third element is absent, no indication is made about the direction.

In CDDL:

```
tag38 = #6.38([tag38-ltag, text, ?tag38-direction])
tag38-ltag = text .abnf ("Language-Tag" .det RFC5646)
tag38-direction = &(ltr: false, rtl: true)
```

[RFC5646](#) = '

```
Language-Tag = langtag           ; normal language tags
               / privateuse       ; private use tag
               / legacy           ; legacy tags

langtag       = language
               ["-" script]
               ["-" region]
               *("-" variant)
               *("-" extension)
               ["-" privateuse]

language      = 2*3ALPHA          ; shortest ISO 639 code
               ["-" extlang]      ; sometimes followed by
                                   ; extended language subtags
               / 4ALPHA           ; or reserved for future use
               / 5*8ALPHA         ; or registered language subtag

extlang       = 3ALPHA            ; selected ISO 639 codes
               *2("-" 3ALPHA)     ; permanently reserved

script        = 4ALPHA            ; ISO 15924 code

region        = 2ALPHA            ; ISO 3166-1 code
               / 3DIGIT           ; UN M.49 code

variant       = 5*8alphanum       ; registered variants
               / (DIGIT 3alphanum)
```

```

extension      = singleton 1*("-" (2*8alphanum))

                                     ; Single alphanumerics
                                     ; "x" reserved for private use
singleton      = DIGIT               ; 0 - 9
                  / %x41-57           ; A - W
                  / %x59-5A           ; Y - Z
                  / %x61-77           ; a - w
                  / %x79-7A           ; y - z

privateuse     = "x" 1*("-" (1*8alphanum))

legacy         = irregular / regular ; different word in RFC

irregular      = "en-GB-oed" / "i-ami" / "i-bnn" / "i-default" /
                  "i-enochian" / "i-hak" / "i-klingon" / "i-lux" /
                  "i-mingo" / "i-navajo" / "i-pwn" / "i-tao" / "i-tay" /
                  "i-tsu" / "sgn-BE-FR" / "sgn-BE-NL" / "sgn-CH-DE"

regular        = "art-lojban" / "cel-gaulish" / "no-bok" / "no-nyn" /
                  "zh-guoyu" / "zh-hakka" / "zh-min" / "zh-min-nan" /
                  "zh-xiang"

alphanum       = (ALPHA / DIGIT)      ; letters and numbers

ALPHA          = %x41-5A / %x61-7A    ; A-Z / a-z
DIGIT          = %x30-39              ; 0-9

```

NOTE: Language tags of any combination of case are allowed. But [section 2.1.1 of \[RFC5646\]](#), part of Best Current Practice 47, recommends a case combination for language tags, that encoders that support tag 38 may wish to follow when generating language tags.

Data items with tag 38 that do not meet the criteria above are invalid (see [Section 5.3.2 of \[STD94\]](#)).

NOTE: The Unicode Standard [[Unicode-14.0.0](#)] includes a set of characters designed for tagging text (including language tagging), in the range U+E0000 to U+E007F. Although many applications, including RDF, do not disallow these characters in text strings, the Unicode Consortium has deprecated these characters and recommends annotating language via a higher-level protocol instead. See the section "Deprecated Tag Characters" in Section 23.9 of [[Unicode-14.0.0](#)].

[A.3.](#) Examples

Examples in this section are given in CBOR diagnostic mode, and then as a pretty-printed hexadecimal representation of the encoded item.

The following example shows how the English-language string "Hello" is represented.

```
38(["en", "Hello"])
```

```
D8 26          # tag(38)
  82          # array(2)
    62        # text(2)
      656E    # "en"
    65        # text(5)
      48656C6C6F # "Hello"
```

The following example shows how the French-language string "Bonjour" is represented.

```
38(["fr", "Bonjour"])
```

```
D8 26          # tag(38)
  82          # array(2)
    62        # text(2)
      6672    # "fr"
    67        # text(7)
      426F6E6A6F7572 # "Bonjour"
```

The following example shows how the Hebrew-language string "שלום" (HEBREW LETTER SHIN, HEBREW LETTER LAMED, HEBREW LETTER VAV, HEBREW LETTER FINAL MEM, U+05E9 U+05DC U+05D5 U+05DD) is represented. Note the rtl direction expressed by setting the third element in the array to "true".

```
38(["he", "שלום", true])
```

```
D8 26          # tag(38)
  83          # array(3)
    62        # text(2)
```

```

6865          # "he"
68          # text(8)
D7A9D79CD795D79D # "שלום"
F5          # primitive(21)

```

[Appendix B](#). Interworking with [RFC 7807](#)

On certain occasions, it will be necessary to carry ("tunnel") [\[RFC7807\]](#) problem details in a Concise Problem Details item.

This appendix defines a Custom Problem Details entry for that purpose. This is assigned Custom Problem Detail key 7807 in [Section 5.2](#). Its structure is:

```

tunnel-7807 = {
  ? &(type: 0) => ~uri
  ? &(status: 1) => 0..999
  * text => any
}

```

To carry an [\[RFC7807\]](#) problem details JSON object in a Concise Problem Details item, first convert the JSON object to CBOR as per [Section 6.2 of \[STD94\]](#). Create an empty Concise Problem Details data item.

Move the values for "title", "detail", and "instance", if present, from the [\[RFC7807\]](#) problem details to the equivalent Standard Problem Detail entries. Create a Custom Problem Detail entry with key 7807. Move the values for "type" and "status", if present, to the equivalent keys 0 and 1 of the Custom Problem Detail entry. Move all remaining key/value pairs (additional members as per [Section 3.2 of \[RFC7807\]](#)) in the converted [\[RFC7807\]](#) problem details object to the Custom Problem Details map unchanged.

The inverse direction, carrying Concise Problem Details in a Problem Details JSON object requires the additional support provided by [\[I-D.ietf-httpapi-rfc7807bis\]](#), which is planned to create the HTTP Problem Types Registry. A Problem Type can then be registered that

extracts top-level items from the Concise Problem Details item in a similar way to the conversion described above, and which carries the rest of the Concise Problem Details item in an additional member via base64url encoding without padding ([Section 5 of \[RFC4648\]](#)). Details can be defined in a separate document when the work on [\[I-D.ietf-httpapi-rfc7807bis\]](#) is completed.

Acknowledgments

Mark Nottingham and Erik Wilde, authors of [RFC 7807](#). Klaus Hartke and Jaime Jiménez, co-authors of an earlier generation of this specification. Christian Amsüss and Marco Tiloca for review and comments on this document.

Fossati & Bormann	Expires 7 November 2022	[Page 18]
-------------------	-------------------------	-----------

Internet-Draft	CoRE Problem Details	May 2022
----------------	----------------------	----------

For [Appendix A](#), John Cowan and Doug Ewell are also to be acknowledged. The content of an earlier version of this appendix was also discussed in the "apps-discuss at ietf.org" and "ltru at ietf.org" mailing lists.

Contributors

Peter Occil
Email: poccil14 at gmail dot com
URI: <http://peteroupc.github.io/CBOR/>

Peter defined CBOR tag 38, basis of [Appendix A](#).

Authors' Addresses

Thomas Fossati
arm
Email: thomas.fossati@arm.com

Carsten Bormann
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org