Workgroup: Network Working Group
Internet-Draft:
draft-yasskin-wpack-bundled-exchanges-04
Published: 7 March 2021
Intended Status: Standards Track
Expires: 8 September 2021
Authors: J. Yasskin
Google

Web Bundles

Abstract

Web bundles provide a way to bundle up groups of HTTP responses, with the request URLs and content negotiation that produced them, to transmit or store together. They can include multiple top-level resources with one identified as the default by a primaryUrl metadata, provide random access to their component exchanges, and efficiently store 8-bit resources.

Discussion Venues

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

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

Source for this draft and an issue tracker can be found at https://github.com/WICG/webpackage.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

To satisfy the use cases in [I-D.yasskin-wpack-use-cases], this document proposes a new bundling format to group HTTP resources. The format is structured as an initial table of "sections" within the bundle followed by the content of those sections.

1.1. Terminology and Conventions

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.

This specification uses the conventions and terminology defined in the Infra Standard ([INFRA]).

2. Semantics

A bundle is logically a set of HTTP representations (Section 7 of [I-D.ietf-httpbis-semantics]), themselves represented by HTTP response messages (Section 2.1 of [I-D.ietf-httpbis-semantics]). The bundle can include an optional URL identifying the primary resource within the bundle and can include other optional metadata. Particular applications can require that the primary URL and/or other metadata is present.

While the order of the representations is not semantically meaningful, it can significantly affect performance when the bundle is loaded from a network stream.

2.1. Operations

Bundle parsers support two primary operations:

1. They can load the bundle's metadata given a prefix of the bundle.

2. They can find a representation within the bundle given that representation's URL (Section 2.2) and the content-negotiation information that would appear in an HTTP request's headers.

2.2. Naming a representation

Representations within a bundle are named by their Content-Location (Section 7.8 of [I-D.ietf-httpbis-semantics]), which holds a URL. This is also known as the representation's URL.

Multiple representations within a bundle can have the same URL, in which case they are distinguished by the content negotiation information contained in their Variants and Variant-Key headers ([I-D.ietf-httpbis-variants]).

This identifying information for each representation is stored in an index (Section 4.2.1) rather than in that representation's HTTP response message.

3. Expected performance

Bundles can be used in two different situations: they can be loaded from storage that provides O(1) access to any byte within the bundle, or they can be sent across a stream that provides bytes incrementally. An implementation MAY prefer either or both situations and SHOULD provide the following performance characteristics in its preferred situations:

3.1. Random access

To load a resource when seeing a bundle for the first time, the implementation reads O(size of the metadata and resource index) before starting to return bytes of the resource.

TODO: Is big-O notation the right way to express expectations here?

3.2. Streaming

When sending a bundle over a stream, the implementation will need to wait until it has the sizes of all contained resources before starting to send the resource index.

When reading a bundle from a stream, the implementation starts returning bytes of a resource after receiving O(1) bytes of that resource, which comes after the O(# of resources) bytes of the index.

4. Format

4.1. Top-level structure

A bundle is a CBOR array ([CBORbis]) with the following CDDL ([CDDL]) schema:

```
webbundle = [
  magic: h'F0 9F 8C 90 F0 9F 93 A6',
  version: bytes .size 4,
  primary-url: whatwg-url,
  section-lengths: bytes .cbor section-lengths,
  sections: [* any ],
  length: bytes .size 8,  ; Big-endian number of bytes in the bundle.
]

whatwg-url = tstr
```

When serialized, the bundle MUST satisfy the core deterministic encoding requirements from Section 4.2.1 of [CBORbis]. This format does not use floating point values or tags, so this specification does not add any deterministic encoding rules for them. If an item doesn't follow these requirements, or a byte-sequence being decoded

as a CBOR item contains extra bytes, the parser MUST signal an error instead of any data it can extract from that item.

High-level fields in the bundle format are designed to provide their length-in-bytes before the field starts so that a recipient trying to stream a bundle from the network can always wait for a known number of bytes instead of needing to implement a streaming CBOR parser.

The magic number is " " (U+1F310 U+1F4E6) encoded in UTF-8. With the CBOR initial bytes for the array and bytestring, this makes the format identifiable by looking for 8? 48 (in base 16) followed by that UTF-8 encoding. Parsers MUST only check the initial nibble of the initial 8? byte in order to accommodate any future version's change in the number of array elements (up to 15).

The version bytestring MUST be 31 00 00 00 in base 16 (an ASCII "1" followed by 3 0s) for this version of bundles. If the recipient doesn't support the version in this field, it MUST either ignore the bundle or fetch and use the content of the primary-url field instead.

The primary-url field identifies both a fallback when the recipient doesn't understand the bundle and a default resource inside the bundle to use when the recipient doesn't have more specific instructions. This field MAY be an empty string, although protocols using bundles MAY themselves forbid that empty value.

The section-lengths and sections arrays contain the actual content of the bundle and are defined in Section 4.2. The section-lengths array is embedded in a byte string to facilitate reading it from a network. This byte string MUST be less than 8192 (8*1024) bytes long, and parsers MUST NOT load any data from a section-lengths item longer than this.

The bundle ends with an 8-byte integer holding the length of the whole bundle.

4.1.1. Trailing length

A bundle ends with an 8-byte CBOR byte string holding a big-endian integer that represents the byte-length of the whole bundle.

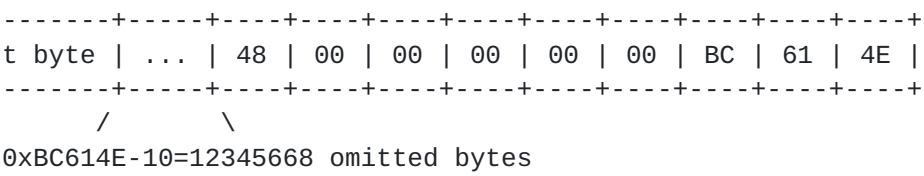

Figure 1: Example trailing bytes

Recipients loading the bundle in a random-access context SHOULD start by reading the last 8 bytes and seeking backwards by that many

bytes to find the start of the bundle, instead of assuming that the start of the file is also the start of the bundle. This allows the bundle to be appended to another format such as a generic self-extracting executable.

4.1.2. Draft version numbers

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

Implementations of drafts of this specification MUST NOT use a version string of 31 00 00 00 (base 16). They MUST instead define an implementation-specific 4-byte string starting with 62 (b) to identify which draft is implemented.

4.2. Bundle sections

A bundle's content is in a series of sections, which can be accessed randomly using the information in the section-lengths CBOR item:

```
section-lengths = [* (section-name: tstr, length: uint) ],
```

This field lists the named sections in the bundle in the order they appear, with each section name followed by the length in bytes of the corresponding CBOR item in the sections array. This allows a random-access parser (Section 3) to jump directly to the section it needs. This specification defines the following sections:

- "index" (Section 4.2.1)
- "manifest" (Section 4.2.2)
- "critical" (Section 4.2.3)
- "responses" (Section 4.3)

Future specifications can register new section names as described in Section 6.2, in order to extend the format without incrementing its version number.

The "responses" section MUST appear after the other three sections defined here, and parsers MUST NOT load any data if that is not the case.

The sections array contains the sections' content. The length of this array MUST be exactly half the length of the section-lengths array, and parsers MUST NOT load any data if that is not the case.

The bundle MUST contain the "index" and "responses" sections. All other sections are optional.

4.2.1. The index section

```
index = {* whatwg-url => [ variants-value, +location-in-responses ] }
variants-value = bstr
location-in-responses = (offset: uint, length: uint)
```

The "index" section defines the set of HTTP representations in the bundle and identifies their locations in the "responses" section. It consists of a CBOR map whose keys are the URLs of the representations in the bundle (Section 2.2). The value of an index entry is an array whose first item is a Variants header field value ([I-D.ietf-httpbis-variants]) or the empty string. This is followed by a sequence of offset/length pairs, one for each representation of this resource. The offset is relative to the start of the "responses" section, with an offset of 0 referring to the head of the CBOR responses array itself. The length is the length in bytes of the response CBOR item holding this representation (Section 4.3).

If the first item in the value of an index entry is empty, it MUST be followed by exactly one offset/length pair. This means there is a single representation for this resource, with no content negotiation.

Otherwise, the first item MUST be followed by one offset/length pair for each of the possible combinations of available-values within the Variants value (the first item of the array) in lexicographic (row-major) order.

For example, given a Variants value of accept-encoding=(gzip br), accept-language=(en fr ja), the list of offset/length pairs will correspond to the Variant-Keys:

- *(gzip en)
- *(gzip fr)
- *(gzip ja)
- *(br en)
- *(br fr)
- *(br ja)

The order of variant-axes is important. If the Variants value were accept-language=(en fr ja), accept-encoding=(gzip br) instead, the location-in-responses pairs would instead correspond to:

- *(en gzip)
- *(en br)

```
    *(fr gzip)

    *(fr br)

    *(ja gzip)

    *(ja br)
```

If the wrong number of offset/length pairs is present in a resource's array, the entire index MUST fail to parse.

A combination of available-values that is omitted from the bundle MUST be signaled by setting its offset and length to 0.

4.2.2. The manifest section

```
manifest = whatwg-url
```

The "manifest" section records a single URL identifying the manifest of the bundle. The URL MUST refer to a resource with representations contained in the bundle itself.

The bundle can contain multiple representations at this URL, and the client is expected to content-negotiate for the best one. For example, a client might select the one matching an accept header of application/manifest+json ([appmanifest]) and an accept-language header of es-419.

Many bundles have a choice between identifying their manifest in this section or in their primary resource, especially if that resource is an HTML file. Identifying the manifest in this section can help recipients apply fields in the manifest sooner, for example to show a splash screen before parsing the primary resource.

4.2.3. The critical section

```
critical = [*tstr]
```

The "critical" section consists of the names of sections of the bundle that the client needs to understand in order to load the bundle correctly. Other sections are assumed to be optional.

If the client has not implemented a section named by one of the items in this list, the client MUST fail to parse the bundle as a whole.

4.3. Responses

```
responses = [*response]
response = [headers: bstr .cbor headers, payload: bstr]
headers = {* bstr => bstr}
```

The "responses" section holds the HTTP responses that represent the HTTP representations in the bundle. It consists of a CBOR array of responses, each of which is pointed to by one or more entries in the "index" section (Section 4.2.1).

The length of the headers byte string in a response MUST be less than 524288 (512*1024) bytes, and recipients MUST fail to load a response with longer headers.

When receiving a bundle in a stream, the recipient MAY process the headers before the payload has been received and MAY start processing the beginning of the payload before the end of the payload has been received.

The keys of the headers map MUST consist of lowercase ASCII as described in Section 8.1.2 of [RFC7540]. Response pseudo-headers (Section 8.1.2.4 of [RFC7540] are included in this headers map.

Each response's headers MUST include a :status pseudo-header with exactly 3 ASCII decimal digits and MUST NOT include any other pseudo-headers.

If a response's payload is not empty, its headers MUST include a Content-Type header (Section 7.4 of [I-D.ietf-httpbis-semantics]). The client MUST interpret the following payload as this specified media type instead of trying to sniff a media type from the bytes of the payload, for example by appending an artificial X-Content-Type-Options: nosniff header field ([FETCH]) to downstream protocols.

4.4. Serving constraints

When served over HTTP, a response containing an application/webbundle payload MUST include at least the following response header fields, to reduce content sniffing vulnerabilities (Section 5.2):

- *Content-Type: application/webbundle

- *X-Content-Type-Options: nosniff

5. Security Considerations

5.1. Version skew

Bundles currently have no mechanism for ensuring that any signed exchanges they contain constitute a consistent version of those

resources. Even if a website never has a security vulnerability when resources are fetched at a single time, an attacker might be able to combine a set of resources pulled from different versions of the website to build a vulnerable site. While the vulnerable site could have occurred by chance on a client's machine due to normal HTTP caching, bundling allows an attacker to guarantee that it happens. Future work in this specification might allow a bundle to constrain its resources to come from a consistent version.

5.2. Content sniffing

While modern browsers tend to trust the Content-Type header sent with a resource, especially when accompanied by X-Content-Type-Options: nosniff, plugins will sometimes search for executable content buried inside a resource and execute it in the context of the origin that served the resource, leading to XSS vulnerabilities. For example, some PDF reader plugins look for %PDF anywhere in the first 1kB and execute the code that follows it.

The application/webbundle format defined above includes URLs and request headers early in the format, which an attacker could use to cause these plugins to sniff a bad content type.

To avoid vulnerabilities, in addition to the response header requirements in Section 4.4, servers are advised to only serve an application/webbundle resource from a domain if it would also be safe for that domain to serve the bundle's content directly, and to follow at least one of the following strategies:

1. Only serve bundles from dedicated domains that don't have access to sensitive cookies or user storage.

2. Generate bundles "offline", that is, in response to a trusted author submitting content or existing signatures reaching a certain age, rather than in response to untrusted-reader queries.

3. Do all of:

    1. If the bundle's contained URLs (e.g. in the manifest and index) are derived from the request for the bundle, percent-encode ([URL]) any bytes that are greater than 0x7E or are not URL code points ([URL]) in these URLs. It is particularly important to make sure no unescaped nulls (0x00) or angle brackets (0x3C and 0x3E) appear.

    2. Similarly, if the request headers for any contained resource are based on the headers sent while requesting the bundle, only include request header field names **and values** that appear in a static allowlist. Keep the set of allowed request header fields smaller than 24 elements to prevent attackers from controlling a whole CBOR length byte.

3. Restrict the number of items a request can direct the server to include in a bundle to less than 12, again to prevent attackers from controlling a whole CBOR length byte.

4. Do not reflect request header fields into the set of response headers.

If the server serves responses that are written by a potential attacker but then escaped, the application/webbundle format allows the attacker to use the length of the response to control a few bytes before the start of the response. Any existing mechanisms that prevent polyglot documents probably keep working in the face of this new attack, but we don't have a guarantee of that.

To encourage servers to include the X-Content-Type-Options: nosniff header field, clients SHOULD reject bundles served without it.

6. IANA considerations

6.1. Internet Media Type Registration

IANA is requested to register the MIME media type ([IANA.media-types]) for web bundles, application/webbundle, as follows:

*Type name: application

*Subtype name: webbundle

*Required parameters:

-v: A string denoting the version of the file format. ([RFC5234] ABNF: version = 1*(DIGIT/%x61-7A)) The version defined in this specification is 1.

Note: RFC EDITOR PLEASE DELETE THIS NOTE; Implementations of drafts of this specification MUST NOT use simple integers to describe their versions, and MUST instead define implementation-specific strings to identify which draft is implemented.

*Optional parameters: N/A

*Encoding considerations: binary

*Security considerations: See Section 5 of this document.

*Interoperability considerations: N/A

*Published specification: This document

*Applications that use this media type: None yet, but it is expected that web browsers will use this format.

*Fragment identifier considerations: N/A

*Additional information:

-Deprecated alias names for this type: N/A

-Magic number(s): 86 48 F0 9F 8C 90 F0 9F 93 A6

-File extension(s): .wbn

-Macintosh file type code(s): N/A

*Person & email address to contact for further information: See the Author's Address section of this specification.

*Intended usage: COMMON

*Restrictions on usage: N/A

*Author: See the Author's Address section of this specification.

*Change controller: The IESG iesg@ietf.org

*Provisional registration? Yes.

6.2. Web Bundle Section Name Registry

IANA is directed to create a new registry with the following attributes:

Name: Web Bundle Section Names

Review Process: Specification Required

Initial Assignments:

Section Name	Specification
"index"	Section 4.2.1
"manifest"	Section 4.2.2
"critical"	Section 4.2.3
"responses"	Section 4.3

Table 1

Requirements on new assignments:

Section Names MUST be encoded in UTF-8.

A section's specification MAY say that, if it is present, another section is not processed.

7. References

7.1. Normative References

[appmanifest]
Caceres, M., Christiansen, K., Lamouri, M., Kostiainen, A., Dolin, R., and M. Giuca, "Web App Manifest", World Wide Web Consortium WD WD-appmanifest-20180523, 23 May 2018, <https://www.w3.org/TR/2018/WD-appmanifest-20180523>.

[CBORbis] 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>.

[CDDL] 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>.

[FETCH] WHATWG, "Fetch", March 2021, <https://fetch.spec.whatwg.org/>.

[I-D.ietf-httpbis-semantics]
Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantics-14, 12 January 2021, <https://tools.ietf.org/html/draft-ietf-httpbis-semantics-14>.

[I-D.ietf-httpbis-variants]
Nottingham, M., "HTTP Representation Variants", Work in Progress, Internet-Draft, draft-ietf-httpbis-variants-06, 3 November 2019, <https://tools.ietf.org/html/draft-ietf-httpbis-variants-06>.

[IANA.media-types] IANA, "Media Types", <http://www.iana.org/assignments/media-types>.

[INFRA] WHATWG, "Infra", March 2021, <https://infra.spec.whatwg.org/>.

[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>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/rfc/rfc7540>.

[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>.

[URL] WHATWG, "URL", March 2021, <https://url.spec.whatwg.org/>.

7.2. Informative References

[I-D.yasskin-wpack-use-cases]
Yasskin, J., "Use Cases and Requirements for Web Packages", Work in Progress, Internet-Draft, draft-yasskin-wpack-use-cases-01, 27 July 2020, <https://tools.ietf.org/html/draft-yasskin-wpack-use-cases-01>.

Appendix A. Change Log

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

draft-04

- Rewrite to be more declarative and less algorithmic.
- Make a bundle represent a set of HTTP Representations, with the Content-Location replacing what was the request URL, and the Variants information, as before, driving content negotiation.
- Make the primary URL optional.
- Remove the signatures section.
- Update Variants examples for the latest Variants draft.
- Removed the distinction between "metadata" and non-metadata sections.

draft-03

- Make the manifest optional.
- Update the reference to draft-yasskin-wpack-use-cases.
- Retitle to "web bundles".

draft-02

- Fix the initial bytes of the format.
- Allow empty responses to omit their content type.
- Provisionally register application/webbundle.

draft-01

- Include only section lengths in the section index, requiring sections to be listed in order.

- Have the "index" section map URLs to sets of responses negotiated using the Variants system ([I-D.ietf-httpbis-variants]).

- Require the "manifest" to be embedded into the bundle.

- Add a content sniffing security consideration.

- Add a version string to the format and its mime type.

- Add a fallback URL in a fixed location in the format, and use that fallback URL as the primary URL of the bundle.

- Add a "signatures" section to let authorities (like domain-trusted X.509 certificates) vouch for subsets of a bundle.

- Use the CBORbis "deterministic encoding" requirements instead of "canonicalization" requirements.

Appendix B. Acknowledgements

Thanks to the Chrome loading team, especially Kinuko Yasuda and Kouhei Ueno for making the format work well when streamed.

Author's Address

Jeffrey Yasskin
Google

Email: jyasskin@chromium.org