

wpack
Internet-Draft
Updates: 6454 (if approved)
Intended status: Standards Track
Expires: 10 September 2020

M. Thomson
Mozilla
9 March 2020

Content-Based Origins for the Web
draft-thomson-wpack-content-origin-00

Abstract

Making content available to clients that are unable or unwilling to contact a web origin enables new means of acquiring content. This document describes a method for taking application state accumulated by an offline user agent in relation to a piece of content and making that state available in a fully online context. This enables continuous use of content, starting from a state where the user agent does not contact an origin and ending with

This document proposes an update to the definition of Origin in RFC 6454. It also proposes changes that would affect HTML, which is outside of the remit of the IETF.

Note to Readers

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/> (<https://mailarchive.ietf.org/arch/browse/wpack/>).

Source for this draft and an issue tracker can be found at <https://github.com/martinthomson/wpack-content> (<https://github.com/martinthomson/wpack-content>).

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 10 September 2020.

Copyright Notice

Copyright (c) 2020 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions and Definitions	4
3. Overview and Comparisons	4
3.1. Offline-to-Online Transition for Content-Based Origins	4
3.2. Offline-to-Online Transition for Signed Exchanges	5
3.3. Comparison with Signed Exchanges	7
3.4. Comparison with Unsigned Bundles	7
4. Content-Based Origin Definition	8
4.1. Content Type Malleability	9
4.2. Hash Agility	9
5. Transfer to HTTPS Origin	10
5.1. Successful Transfer	10
5.2. Unsuccessful Transfer	11
5.3. Transfer Target	12
5.4. State Transfer	12
5.4.1. Transfer of Content	13
5.4.2. Transfer of Storage	13
5.4.3. Transfer of Permissions	14
5.5. Navigation Transfers	14
5.6. Communication Between Origins	15
6. Security Considerations	15
7. IANA Considerations	15
8. References	15
8.1. Normative References	15
8.2. Informative References	16
Acknowledgments	18
Author's Address	18

1. Introduction

The web relies on the concept of origins [ORIGIN] as the primary means of identification for resources. A strongly authenticated HTTPS origin is the basis for many security decisions. However, establishing this identity relies on having an active TLS [TLS] connection to a server that is considered authoritative for the origin; see Section 11 of [HTTP].

A user agent that is offline when content is received cannot establish this authority. Similarly, a user agent might be unwilling to contact a server at the time that content is received. One reason for not contacting a server might be to avoid leaking information about the context in which content was referenced. In either case, content cannot be attributed to the origin at the time it is received.

In operation, content might be delivered by any means other than a TLS connection to the intended server. Then there might be a period during which the content is used. For instance, the content might be a web page that is capable of functioning offline, though certain functions might be unavailable.

At some after content is received, the user agent might want to establish a connection to the server and continue use. True continuity of use could depend on state established during the period that the user agent did not contact the server.

This document proposes a design whereby content can be ascribed an identity that is based solely on that content. Together with a means of transferring control over state established by one origin to another, this allows content to be delivered offline and used with the ability to transition to a full online interaction with a web server.

Content-based origins are proposed as an alternative to signed exchanges [SXG], which ascribe an HTTPS origin to content through the use of signatures. Signed exchanges depends on finding a way to modify the core concept of web origins that allows for object-based authority. Signed exchanges avoid any problems arising from transfer of state between origins. However, a fundamental change to the way in which authority is determined requires the use of a number of safeguards. These safeguards contain both technical mechanisms and usage constraints. These constraints could be operationally challenging to meet, but violating them could have consequences for security.

2. Conventions and Definitions

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 document uses the term "user agent" consistent with the usage in both [HTTP] and [HTML]. Colloquially, "user agent" is a web browser.

3. Overview and Comparisons

This section provides an overview of how content-based origins might be used and compares that to designs based on signed exchanges.

The most interesting scenario involves the transition from an state where the target origin has not been contacted (that is, the user agent is either offline or effectively so), to one where the user agent contacts the target origin (the user agent is online).

3.1. Offline-to-Online Transition for Content-Based Origins

For a content-based origin, content is loaded from a bundle. After loading the bundle, the browser treats content in the bundle as existing in an origin unique to the content of the bundle; see Section 4.

If the bundle contains a transfer target URL (see Section 5.3), the browser then attempts to load that resource, providing an indication to the target origin that a transfer from the content-based origin is possible. If the origin accepts the transfer of state, state is transferred from the content-based origin to the target origin. This sequence is shown in Figure 1.

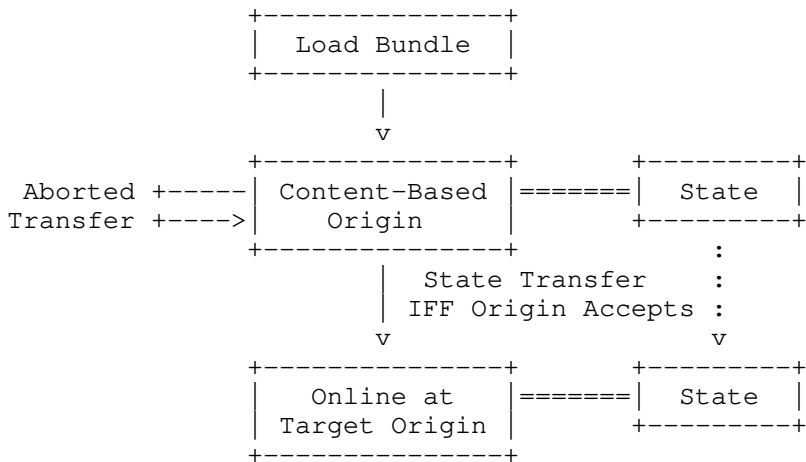


Figure 1: State Transfer for Content-Based Origins

Failure to connect to the target origin - such as when the browser has no active Internet connection - causes the redirect to be aborted. The browser continues to display the bundle content.

If the origin does not accept the transfer, the browser shows content from the target origin and any state from the bundle is destroyed.

3.2. Offline-to-Online Transition for Signed Exchanges

With [LOADING] and [SXG], content is loaded from a signed bundle. After loading the bundle, if the signature is considered valid, the browser stores bundled content in a cache that is specific to the target origin. The browser then redirects to the request (or target) URL from the bundle. The browser uses content from the bundle in place of fetching from the origin. This is illustrated in Figure 2.

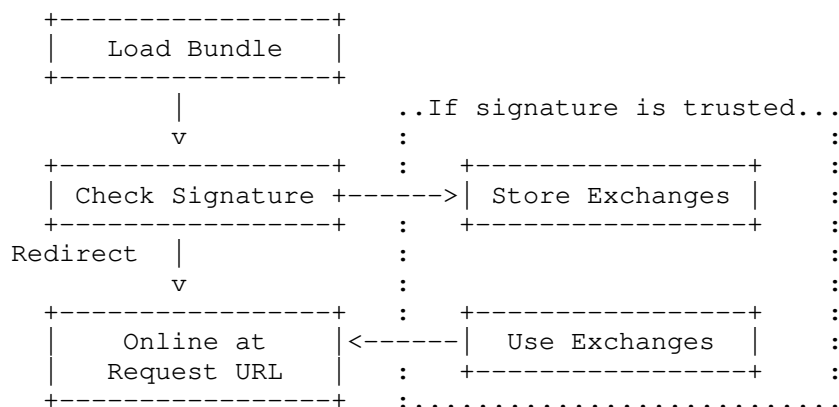


Figure 2: Content Use with Signed Exchanges

Having signed content from the bundle allows use of that content prior to connecting to the origin. Importantly, it allows attribution of any operations to that origin. Optimizations that otherwise might not be possible are enabled because resources from the bundle can be treated as if they were loaded from the target origin, but the browser does not need to make a request to the origin.

Critically, if the browser is fully offline, it can decide to operate with the bundle without having to connect to the origin. If the bundled content is signed and trusted, the application to operate offline. Any actions performed act on state for the target origin.

Note: While Service Workers [SERVICE-WORKERS] can offer a similar experience, they require that the browser be online initially to load the service worker script. This design requires no prior interaction with the target origin.

Relative to content-based origins, the use of signatures avoids the need to connect to the target origin and confirm knowledge of the content. For cases where the transition to online status is intended to be immediate, signed exchanges allow a redirect to happen without any requests being made. A signed exchange will therefore display content that is attributed to the target origin faster.

Failure to validate a signature causes a redirect to an online location. If the client is offline, that origin will be inaccessible. If the client is online, none of the optimizations afforded by the bundle are available and this appears to be a normal redirection.

[[Note that it is unclear whether a browser might be able to fall back to treating a signed bundle as unsigned if the signature is bad.]]

3.3. Comparison with Signed Exchanges

Signed exchanges attempt to address the problem of attributing content to an origin. In effect, they add an object-based security model to the existing channel-based model used on the web. Signatures over bundles (or parts thereof) are used by an origin to attest to the contents of a bundle.

Having two security models operate in the same space potentially creates an exposure to the worst properties of each model. To reduce the chances that the drawbacks of the newly-added object-based model affect existing channel-based usages, signed exchanges include a number of hardening measures. In addition to signatures, there are required modifications to certificates, constraints on validity periods, and a range of limitations on the types of content that can be signed.

In comparison, content-based origins do not require signatures. Questions of validity only apply at the point that a state transfer is attempted. However, this has drawbacks also. Content is not attributed to origins and state is not available to an online origin until a transfer is complete. This avoids the complexity inherent to merging two different security models, but the process of state transfer could be quite complicated in practice.

In terms of usability, the identity attributed to content from a content-based origin is opaque and not particularly relatable. Though state might eventually be adopted by some origin, communicating the true status of content could be challenging. Finally, content-based origins aren't prevented from interacting with HTTP origins, which could lead to surprising outcomes if existing code is poorly unprepared for this possibility.

3.4. Comparison with Unsigned Bundles

Unsigned bundles [UNSIGNED] proposes a model whereby bundles served by a given origin could be declared to be part of that origin and trusted. Unsigned bundles would otherwise be considered untrusted and would be isolated in some fashion. This is similar to the way in which a content-based origin might be transferred to a target origin.

That proposal divides bundles into trusted and untrusted bundles. Trusted bundles would be able to access state from the origin that served them.

In comparison, content-based origins would always be used for bundles and the distinction between trusted and untrusted is unnecessary. Content from the bundle would be usable to a target origin that accepts a state transfer.

4. Content-Based Origin Definition

A content-based origin ascribes an identity to content based on the content itself. For instance, a web bundle [BUNDLE] is assigned a URI based on its content alone.

The sequence of bytes that comprises the content or bundled content is hashed using a hash function that is strongly resistant to collision and pre-image attack, such as SHA-256 [SHA-2]. The resulting hash is encoded using the Base 64 encoding with an URL and filename safe alphabet [BASE64].

This can be formed into the ASCII or Unicode serialization of an origin based on the Named Information URI scheme [NI]. This URI is formed from the string "ni://", the identifier for the hash algorithm (see Section 9.4 of [NI]); a semi-colon (";"), and the base64url encoding of the hash function output. Though this uses the ni URL form, the authority and query strings are omitted from this serialization.

For instance, the origin of content comprising the single ASCII character 'a' is represented as "ni:///sha-256;ypeBEsobvcr6wjGzmiPcTaeG7_gUfE5yuYB3ha_uSLs".

In tuple form, the origin is comprised of the scheme ("ni") and a host equal to the concatenation of hash algorithm identifier, semi-colon, and base64url-encoded hash function output. The port number is always absent for an origin in this form.

Design Note: This design currently assumes that there won't be a hash-based URI scheme developed for bundles. It would be vastly preferable to have a URI scheme for bundled content. It would then be possible to reference content in a bundle from outside the bundle, and internal references could be canonicalized. Importantly, state transfer could also be initiated toward another bundle. That could be used to upgrade bundles and other nice features.

This definition of origin for named information ("ni://") URIs extends the definition of origin in Section 4 of [ORIGIN].

4.1. Content Type Malleability

The hash used to generate this origin does not include any indication of content type or the source of information. If it were possible for the same content to be interpreted as valid instances of multiple different content types, that might be exploited by an attacker.

A user agent SHOULD enact a policy of only attributing a content-based origin to content that is unambiguously interpreted in a single form. If it were possible for content to be interpreted as valid for multiple content types and those could be attributed to the same content-based origin, that might be exploited by an attacker. A sample policy might only allow web bundles and HTML files to be assigned a content-based origin; as the first bytes of these formats are unambiguously different, this ensures that the same content to be interpreted as valid for both content types.

4.2. Hash Agility

This design depends to some extent on the hash algorithm remaining constant during the time that the content-based origin is used. A change in hash algorithm changes the identity of the resource.

It is possible to transfer state from a content-based origin derived using one hash algorithm to another without affecting the content of the origin itself. It is not often that content depends on knowing its own identity. However, the identity of the origin might be made visible to other origins. For instance, the `window.postMessage` API [HTML] allows content to target a specific origin for sending messages and to identify the source origin of incoming messages.

For the purposes of determining equality, user agents might consider hashes of the same content with different hash algorithms to be equal. For instance, a user agent might consider "sha-256:ypeBEsobvcr6wjGzmiPcTaeG7_gUfE5yuYB3ha_uSLs" to be equal to "sha-384;VKWbnyKwuAia2EJ-Vit8I6vYc0huHwNdzpzWl-hRdQM8qojm1XvDXvrgta_TFF8x". This requires that this equivalence is known to the user agent.

Of the hash algorithms defined in [NI], only "sha-256" is permitted for use with content-based origins. This usage has no need for a truncated hash as the value does not usually need to be manually entered. Furthermore, the use of multiple hash algorithms introduces complexity.

5. Transfer to HTTPS Origin

In order to transfer state to a target origin, the server for that origin needs to be contacted. Initiating transfer likely requires that an API be defined for use by the content of the bundle. Transfer might be automatically initiated when navigating to a URL from a bundle; see Section 5.5.

A transfer is only possible where a transfer target URL is known for content; see Section 5.3.

After a state transfer is initiated, the user agent fetches the transfer target URL. The source content is hashed twice; once as described in Section 4; then the resulting value is hashed again. The twice-hashed value is encoded into a Sec-Content-Origin header field and added to the request for the transfer target URL. If a Sec-Content-Origin header field in the response contains the hash of the content (that is, the pre-image of the value in the request), that indicates that the server is both willing to receive the transfer and that the server knows the content.

The value of the Sec-Content-Origin header field is expressed as a structured header [SH] dictionary with keys being the hash algorithm identifier (from [NI]) and byte sequence values of the hash.

Multiple values can be provided with different hash algorithm identifiers. The values from the response correspond to the values in the request with the same hash algorithm identifier. For the transfer to be successful, clients **MUST** validate that at least one value in the response hashes to the corresponding value in the request; clients **SHOULD** validate all provided values. If the content does not hash to the any provided value, the transfer is unsuccessful.

Note: The response to a state transfer can be served from cache.

5.1. Successful Transfer

If the transfer is successful, state associated with the content-based origin will be transferred to the target origin. The user agent **MAY** either remember the identity of the content-based origin and consider any content-based origin to be equal to the transferred origin.

For example, assuming that a single 'a' character is valid content, then the client would include the following in a request (line breaks are added to examples for formatting reasons):

Sec-Content-Origin:

sha-256=:v106/7c+/S7Gw2rTES3ZM+/tY8Thy//PqI4nWcFE8tg=:

A successful state transfer would occur if the response from the server included:

Sec-Content-Origin:

sha-256=:ypeBEsobvcr6wjGzmiPcTaeG7/gUfE5yuYB3ha/uSLs=:

A user agent that automatically follows redirections (3xx status codes) MUST allow the server to redirect to a resource that provides the response. Redirection can change the origin that ultimately accepts the transfer. Any redirection to an origin that is not strongly authenticated MUST cause the transfer to fail.

After a successful transfer, the user agent MAY treat the content-based origin as an alias for the origin to which state was transferred. This aliasing might be particularly useful in addressing hash agility; see Section 4.2.

5.2. Unsuccessful Transfer

A transfer can be unsuccessful in two ways. If the target origin cannot be contacted successfully, the content-based origin continues to exist. Any API might indicate that the attempt failed. Failure to transfer state is expected when the user agent is offline. After failing to contact the target origin, a transfer can be attempted at a later time. This causes navigation to fail and the user agent MAY display a URL from the content-based origin.

[[TODO: Again, this requires that we define a means of identification for content inside bundles.]]

A valid, final HTTP response that indicates anything other than a server error (that is, 5xx status codes) or lack of authority (the 421 status code [HTTP2]) without including the hash pre-image in the response MUST be treated as a failed migration. After a failed migration, information about the target origin SHOULD be removed from interfaces related to the content-based origin, except for diagnostic purposes. The content-based origin MAY continue to exist but further attempts to transfer state MUST immediately fail.

After a valid HTTP response, the user agent navigates to the transfer target URL or any resource that was included in any redirections.

Any valid HTTP response, successful or not, MUST cause data associated with the content-based origin to be cleared as though clear-site-data were invoked [CLEAR-DATA].

5.3. Transfer Target

To enable transfer, content **MUST** include an attribute that indicates the transfer target URL. If a bundle could contain multiple potential entry points, each entry point for the bundle would separately specify a different transfer target URL.

[[Intuitively, it seems prudent to limit transfer target URLs in the same bundle to the same origin, but I don't have a concrete reason for doing so right now.]]

A transfer target URL does not need to be specified if the intent is to never support the ability to transfer to an online state.

The origin of the transfer target URL is the target origin. Only the target origin can be the target of a state transfer. After a successful transfer, the user agent loads the transfer target URL.

Note: Including the transfer target URL in content means that the content hash is dependent on its value. This ensures that different content-based origins are produced if different transfer target URLs are used. Two different origins cannot share state.

A commitment in this form could allow user agents to present the target origin in interfaces. While no strong assurances can be made about the attribution of content to this origin, this might make it easier to generate user interfaces.

Different content types might provide a way of encoding the transfer target URL. The URL could be provided external to the content, but this requires some unspecified means of ensuring that the content-based origin depends on the value of the URL; this document only supports content that can encode the transfer target URL.

5.4. State Transfer

Only limited state information can be transferred between origins. This is limited to those items that are identified here, or those that are added in later extensions. This document describes how content (Section 5.4.1), stored data (Section 5.4.2), and permissions (Section 5.4.3) are transferred.

Upon successful completion of transfer, if loading the transfer target resource produces an HTML document, an event is delivered to content of that resource that describes what information was transferred.

A user agent MAY request user permission to transfer some state. This might be conditioned on what state is being transferred. For instance, a user agent might prompt before transferring permissions [PERMISSIONS]; see Section 5.4.3.

5.4.1. Transfer of Content

How content that is delivered in the bundle is used by the target origin is probably the most important aspect of any design in this space. This document proposes that all content in the bundle is adopted by the origin after a transfer completes.

This design means that bundles need a way to describe how all their resources map to URLs in the target origin. If it is possible for bundles to contain content from multiple origins, content from other origins won't be accessible after transfer without first making a request to the other origin.

Content loaded from a bundle MUST NOT be made available for use by origins other than the target origin. For example, if the bundle includes a script that maps to a URL on the target origin and a site at another origin loads that script, then the user agent fetches the script. The user agent MAY use a conditional request with If-None-Digest [IF-DIGEST] and the Digest header field [DIGEST] to reduce the overhead of these requests at its discretion, noting that this might introduce observable timing signals.

Restricting content to use by the target origin avoids the negative effects described in [SRI-ADDRESSABLE]. The target origin is required to demonstrate knowledge of the contents of the bundle, which prevents that origin from being poisoned by an attacker. For origins other than the target origin, content cannot be emplaced through the use of a bundle.

Alternative methods for transfer of content might require per-resource confirmation by the origin. That might be loosened so that resources that use Subresource Integrity [SRI] do not require confirmation. But that increases the need to provide integrity attributes at the point that resources are referred to. If content is fetched programmatically, that might be operationally challenging.

5.4.2. Transfer of Storage

Upon transfer, any indexedDB databases [INDEXEDDB] are transferred to the target origin. The names of any databases are prefixed with the origin from which they came. If a database with the same name exists at the destination, a new name is selected.

The event that indicates transfer will list the databases that were transferred, their names and any name changes.

5.4.3. Transfer of Permissions

As part of a state transfer, any persistent permissions that a user granted the content-based origin might be transferred to the target origin.

Whether any given permission is transferred and what conditions are attached to the transferral will depend on the policy of the user agent. It might be considered best to discard permissions and request each anew as necessary. As transferral is a one-time event, causing prompts to be reinitiated might not be too much of an imposition.

5.5. Navigation Transfers

Initiating transfer on navigation enables pre-loading of content from bundles.

To address this use case, the user agent navigates to a bundle. That bundle might be served from the site providing the link to ensure that information about the linking page is not revealed to the target origin prior to navigation.

Navigating to the bundle causes an immediate load of the content of the bundle. This assumes that there is either a URL component that specifies a single entry point for the bundle or that the bundle itself identifies the entrypoint.

If a transfer target URL is specified for the target resource, the user agent navigates to that URL. The user agent **MUST** fetch the transfer target URL and include a Sec-Content-Origin header field in the request as described in Section 5. This request **MAY** also include the ETag header field from the bundled content.

Using If-None-Digest [IF-DIGEST] allows the resource at the transfer target URL to return a 304 (Not Modified) status code in response to a request if the content provided in the bundle known to the server to be sufficient.

The result is that the state from the bundle is transferred to the target origin. As the navigation is immediate, no state will have been created within the bundle so the transfer of state can be skipped by the browser if navigation from a previously unused content-based origin is successful. It is possible that the content-

based origin might have previously used, in which case state transferrance might occur.

5.6. Communication Between Origins

Without knowledge of the content of a resource, or bundle of resources, a content-based origin will be impossible to guess. This means that communication is only possible if the frame in which the content is loaded by the origin attempting communication, or the content is known to that origin.

6. Security Considerations

This design avoids questions about having to attribute authority to a static bundle of content. Instead, it creates a new origin type and enables the transfer of state from one origin to another. The target origin can decide whether to accept a transfer, thereby avoiding any questions of poisoning of content. If content is found to be compromised, an origin can subsequently refuse to accept a transfer from that content.

Transfer of state to an origin with an "http://" origin is not possible. This mechanism depends on the target origin being strongly authenticated.

7. IANA Considerations

[[TODO: Register the Sec-Content-Origin header field. Probably a lot more.]]

8. References

8.1. Normative References

- [BASE64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [BUNDLE] Yasskin, J., "Bundled HTTP Exchanges", Work in Progress, Internet-Draft, draft-yasskin-wpack-bundled-exchanges-02, 26 September 2019, <<http://www.ietf.org/internet-drafts/draft-yasskin-wpack-bundled-exchanges-02.txt>>.
- [HTTP] Fielding, R., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantics-07, 7 March 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-httpbis-semantics-07.txt>>.

- [HTTP2] Belshé, 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/info/rfc7540>>.
- [NI] Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B., Keranen, A., and P. Hallam-Baker, "Naming Things with Hashes", RFC 6920, DOI 10.17487/RFC6920, April 2013, <<https://www.rfc-editor.org/info/rfc6920>>.
- [ORIGIN] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <<https://www.rfc-editor.org/info/rfc6454>>.
- [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>>.
- [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/info/rfc8174>>.
- [SH] Nottingham, M. and P. Kamp, "Structured Headers for HTTP", Work in Progress, Internet-Draft, draft-ietf-httpbis-header-structure-14, 28 January 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-httpbis-header-structure-14.txt>>.
- [SXG] Yasskin, J., "Signed HTTP Exchanges", Work in Progress, Internet-Draft, draft-yasskin-http-origin-signed-responses-08, 4 November 2019, <<http://www.ietf.org/internet-drafts/draft-yasskin-http-origin-signed-responses-08.txt>>.

8.2. Informative References

- [CLEAR-DATA] West, M., "Clear Site Data", W3C ED, 13 November 2017, <<https://w3c.github.io/webappsec-clear-site-data/>>.
- [DIGEST] Polli, R. and L. Pardue, "Digest Headers", Work in Progress, Internet-Draft, draft-ietf-httpbis-digest-headers-01, 3 November 2019, <<http://www.ietf.org/internet-drafts/draft-ietf-httpbis-digest-headers-01.txt>>.
- [HTML] WhatWG, ., "HTML", WhatWG Living Standard, 4 March 2020, <<https://html.spec.whatwg.org/>>.

- [IF-DIGEST] Thomson, M., "Conditional HTTP Requests Using Digests", Work in Progress, Internet-Draft, draft-thomson-http-if-digest-00, 9 March 2020, <<https://tools.ietf.org/html/draft-thomson-http-if-digest-00>>.
- [INDEXEDDB] Alabbas, A. and J. Bell, "Indexed Database API 3.0", W3C ED, 29 February 2020, <<https://w3c.github.io/IndexedDB/>>.
- [LOADING] Yasskin, J., "Loading Signed Exchanges", 21 February 2020, <<https://wicg.github.io/webpackage/loading.html>>.
- [PERMISSIONS] Lamouri, M., Cáceres, M., and J. Yasskin, "Permissions", W3C ED, 30 October 2019, <<https://w3c.github.io/permissions/>>.
- [SERVICE-WORKERS] Russell, A., Song, J., Archibald, J., and M. Kruisselbrink, "Service Workers 1", W3C ED, 24 February 2020, <<https://w3c.github.io/ServiceWorker/v1/>>.
- [SHA-2] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [SRI] Akhawe, D., Braun, F., Marier, F., and J. Weinberger, "Subresource Integrity", W3C ED, 4 January 2020, <<https://w3c.github.io/webappsec-subresource-integrity/>>.
- [SRI-ADDRESSABLE] Hill, B., "Subresource Integrity Addressable Caching", 15 October 2016, <<https://hillbrad.github.io/sri-addressable-caching/sri-addressable-caching.html>>.
- [TLS] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [UNSIGNED] Yasskin, J., "Navigation to Unsigned Web Bundles", 20 February 2020, <<https://github.com/WICG/webpackage/blob/master/explainers/navigation-to-unsigned-bundles.md>>.

Acknowledgments

This work is hardly original, nor is it an individual effort.
[[TODO: acknowledge.]]

Author's Address

Martin Thomson
Mozilla

Email: mt@lowentropy.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2021

J. Yasskin
Google
7 March 2021

Web Bundles
draft-yasskin-wpack-bundled-exchanges-04

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

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 8 September 2021.

Copyright Notice

Copyright (c) 2021 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology and Conventions	3
2. Semantics	3
2.1. Operations	3
2.2. Naming a representation	3
3. Expected performance	4
3.1. Random access	4
3.2. Streaming	4
4. Format	4
4.1. Top-level structure	4
4.1.1. Trailing length	6
4.1.2. Draft version numbers	6
4.2. Bundle sections	6
4.2.1. The index section	7
4.2.2. The manifest section	9
4.2.3. The critical section	9
4.3. Responses	9
4.4. Serving constraints	10
5. Security Considerations	10
5.1. Version skew	10
5.2. Content sniffing	11
6. IANA considerations	12
6.1. Internet Media Type Registration	12
6.2. Web Bundle Section Name Registry	13
7. References	14
7.1. Normative References	14
7.2. Informative References	15
Appendix A. Change Log	16
Appendix B. Acknowledgements	17
Author's Address	17

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 $O(\text{size of the metadata and resource index})$ 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 $O(\# \text{ of resources})$ 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.

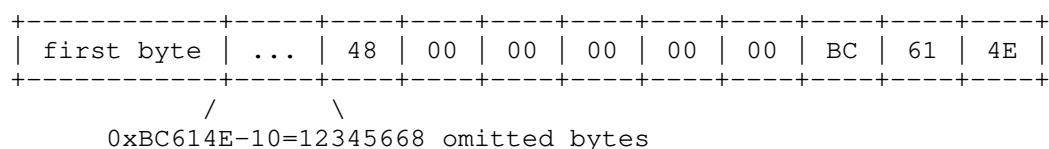


Figure 1: Example trailing bytes

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-Key"s:

- * (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 (<https://url.spec.whatwg.org/#percent-encode>) ([URL]) any bytes that are greater than 0x7E or are not URL code points (<https://url.spec.whatwg.org/#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 (<mailto: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