TLS
Internet-Draft
Intended status: Experimental
Expires: 4 November 2026

S. Farrell
Trinity College Dublin
R. Salz
Akamai Technologies
B. Schwartz
Meta Platforms, Inc.

3 May 2026

A well-known URI for publishing service parameters
draft-ietf-tls-wkech-12

Abstract

We define a well-known URI at which an HTTP origin can inform an authoritative DNS server, or other interested parties, about its Service Bindings. Service binding data can include Encrypted ClientHello (ECH) configurations, that may change frequently. This allows the HTTP origin, in collaboration with DNS infrastructure elements, to publish and rotate its own ECH keys. Other service binding data such as information about TLS supported groups is unlikely to change quickly, but the HTTP origin is much more likely to have accurate information when changes do occur. Service data published via this mechanism is typically available via an HTTPS or SVCB resource record.

Note

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

The source for this draft is in https://github.com/tlswg/wkesni/ Issues and PRs are welcome there too.

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 4 November 2026.

Table of Contents

1. Introduction

The Service Bindings (SVCB) in DNS RFC [RFC9460] defines how to handle information needed to make connections to services by using a new DNS record set (RRset) with the HTTPS or SVCB resource record type. Many HTTP origins are managed by proprietary systems that enable coordinated control of the TLS server configuration and the DNS zone file contents. For example, a single configuration source might be used to set the ECH keys for TLS and populate the DNS zone

file. However, some deployments manage TLS and DNS separately, and require a mechanism like the one defined here to pass information from the TLS server to the DNS zone.

Encrypted ClientHello (ECH) [RFC9849] for TLS1.3 [RFC8446] defines a confidentiality mechanism for server names and other ClientHello content in TLS, and there is a well-defined way [RFC9848] to include Encrypted Client Hello (ECH) configuration information (ECHConfigList values) in HTTPS or SVCB RRsets. Each ECHConfigList value contains a list of the public components of a key pair that will typically be periodically (re-)generated by a TLS server.

This document uses ECH to provide a concrete motivating example, but the mechanism defined here can be used to deliver other kinds of service information about the origin, intended to be published in HTTPS or SVCB RRs. Other use cases, such as the TLS preferred groups service parameter defined in [I-D.ietf-tls-key-share-prediction], can also be supported via this mechanism.

We use the term "zone factory" (ZF) for an entity which can validate service binding data and that has write access to the zone file. We assume the ZF can also make secure HTTP requests to the origin to validate service binding data that the origin has asked to be published. We define a well-known URI [RFC8615] on the origin server that allows the ZF to poll for changes to service binding data.

For example, if a web server generates new ECHConfigList values hourly and publishes those at the well-known URI, the ZF can poll that URI. When the ZF sees new values, it can check if those work, and if they do, then update the zone file and re-publish the zone. If ECH is being operated in split-mode then the web server (backend) can similarly poll via the ECH client-facing server at the well-known URI and then create its own value to publish for the ZF to read. ECH split-mode is defined in [RFC9849] and an example is provided below (Section 3).

2. Terminology

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.

We define or re-use the following terms:

Zone factory (ZF): an entity that has write-access to the authoritative DNS server zone database.

Client-Facing Server (CFS): the web server that has an ECH private value. This processes the outer ClientHello and attempts ECH decryption. The name of client-facing server will typically be the public_name value used in an ECHConfig.

Backend: the web server that will process the inner ClientHello. Note that even if client-facing server and backend are on the same web server, they almost certainly have different DNS names.

Shared-mode: this is where client-facing and backend servers are the same web server.

Split-mode: this refers to the case where the client-facing server only does ECH decryption but the TLS session is between the client and backend, which will typically be on a different host to client-facing server

regeninterval: the number of seconds after which the value retrieved after acessing a well-known URI may be changed.

3. Example uses of the well-known URI for ECH

Some example ECH deployments are outlined here.

3.1. Shared-mode deployment.

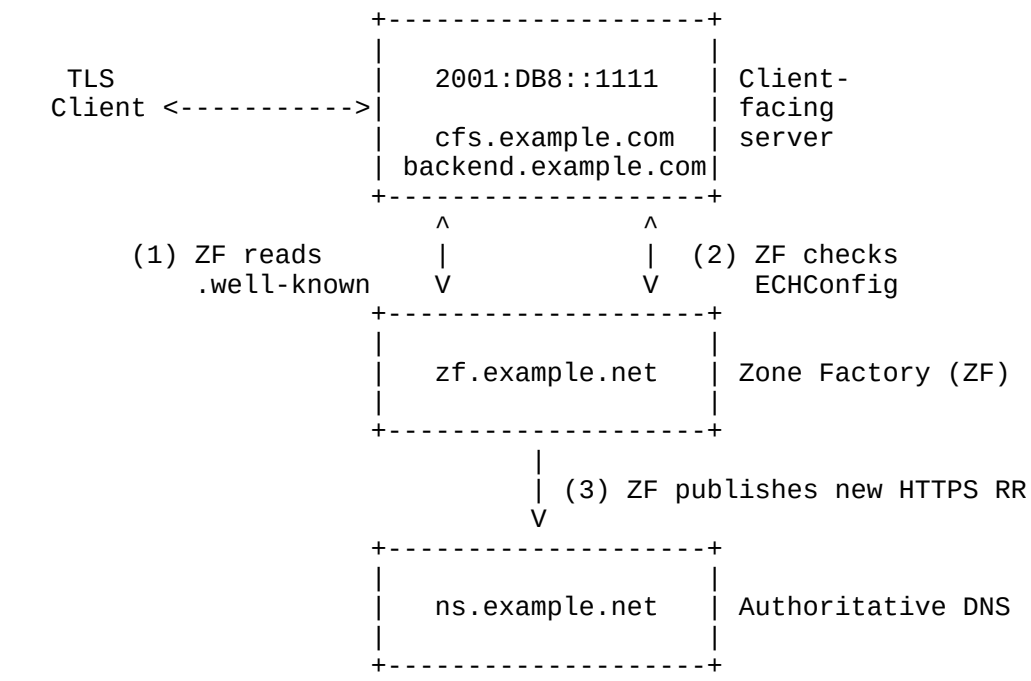

Figure 1: Shared-Mode Topology with Zone Factory and DNS

The shared-mode ECH server generates new ECHConfigList values every "regeninterval" seconds via some regular, automated process (e.g., a cronjob). ECHConfigList values are "current" for an hour, and remain usable for three hours from the time of generation. The automated process updates the ECHConfigList values in a JSON resource (see Figure 5) at the well-known URI, https://backend.example.com/.well-known/origin-svcb.

These steps may then occur:

1. On the ZF, another regularly executed job uses an HTTP client to retrieve this JSON resource from backend.example.com. The data MUST be fetched via HTTPS and the certificate validity MUST be verified.

2. If there is no change to the JSON content, then the ZF is done processing for this backend. If there is a change to the JSON content, the ZF attempts to connect to backend.example.com using these ECH values and confirms that they are working.

3. The ZF observes that the JSON resource has a regeninterval of 3600 seconds, and chooses a DNS TTL of 1800. It updates the DNS zone file for backend.example.com and re-publishes the zone containing the new ECHConfigList values instead of the old.

When "regeninterval" seconds have passed, the ZF attempts to refresh its cached copy of the JSON resource. If the resource has changed, it repeats this process.

The JSON file produced in this example might be as shown in Figure 2, and the resulting HTTPS RR might then be as shown in Figure 3. (New-lines are added for presentation.)

```
{
 "regeninterval" : 3600,
 "endpoints" : [ {
    "priority" : 1,
    "params" : {
      "ech" : "AEL+DQA+ogAgACDzFvDxhHtneEqwlof1omyso8XXzskgR5w\
               wuDxe3EweawAEAAEAAQAPY2ZzLmV4YW1wbGUuY29tAAA="
    }
 }]
}
```

Figure 2: Shared-Mode JSON Example

```
$ dig +short HTTPS backend.example.com
1 . ech=AEL+DQA+ogAgACDzFvDxhHtneEqwlof1omyso8XXzskgR5w\
        wuDxe3EweawAEAAEAAQAPY2ZzLmV4YW1wbGUuY29tAAA=
```

Figure 3: Shared-Mode HTTPS RR

3.2. Split-mode deployment.

In this section, we describe one possible way in which ECH split-mode could be deployed and make use of this .well-known scheme. There are of course various other deployment options that might be more effective.

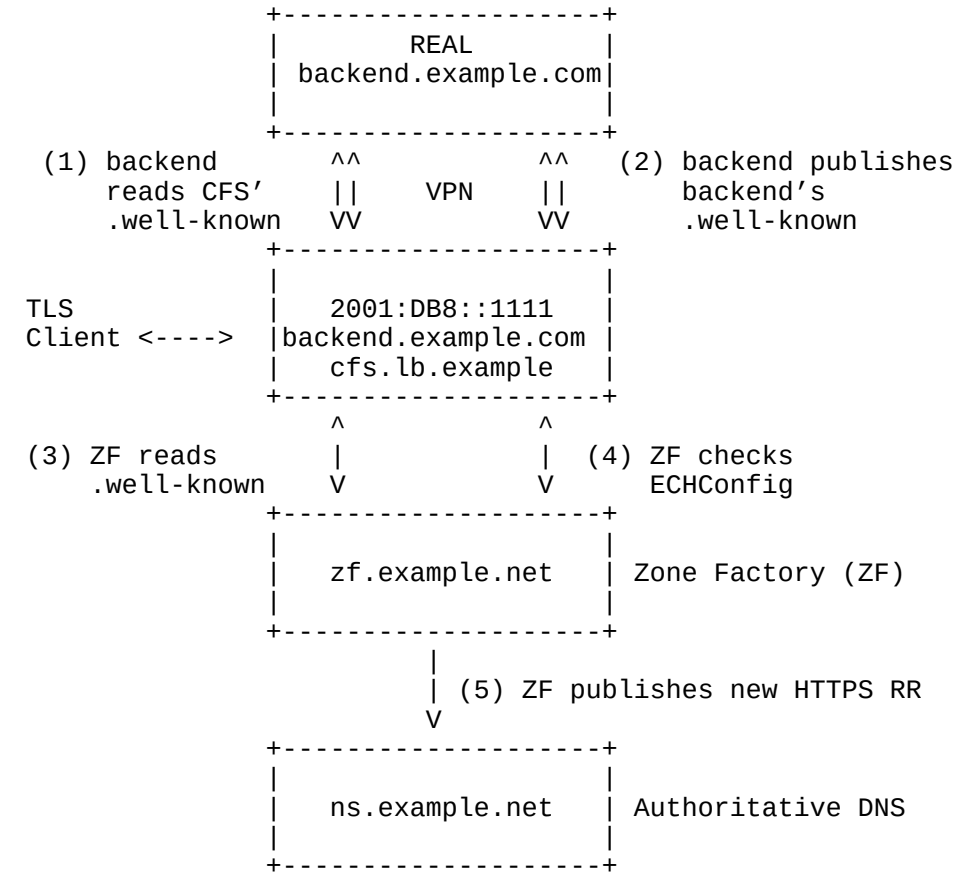

Figure 4: Split-Mode Topology with Zone Factory and DNS

In this topology, the overall process is as in the previous section, but with the following differences:

1. The CFS is a load-balancer and only does ECH decryption. The load-balancer is called cfs.lb.example.

2. Traffic between the CFS and backend is protected using some kind of VPN.

3. The A/AAAA values for backend.example.com and cfs.lb.example are the same.

4. backend.example.com content is hosted on some machine(s) accessible via the CFS and then via the VPN. (In Figure 4 this is shown as "REAL backend.example.com".)

5. backend.example.com wants to acquire the ECH config information from the CFS by querying the CFS' .well-known URL, in this case https://cfs.lb.example/.well-known/origin-svcb

6. backend.example.com then publishes its chosen JSON (presumably including the relevant ECH config information) at https://backend.example.com/.well-known/origin-svcb That resource is hosted on the "REAL" backend.example.com machine.

7. The ZF attempts to connect to https://backend.example.com using the generated HTTPS record(s), including the relevant ECH values, and confirms that all is working before updating the zone.

Note that in this example the ZF does not necessarily know that ECH split-mode is in use.

4. The origin-svcb well-known URI

If backend.example.com wants to convey information to the Zone Factory, it publishes the JSON content defined in Section 5 at: https://backend.example.com/.well-known/origin-svcb

The well-known URI defined here MUST be an "https" URL and therefore the ZF can verify the correct backend is being accessed.

If new service binding data fails to validate (as per Section 6), then the ZF MUST NOT modify the zone.

If the origin desires some particular service binding data be included in an HTTPS or SVCB resource record, then it MUST include the relevant value(s) in the JSON resource published at the well-known URI. A ZF may seperately be configured to add additional information to the published resource records as desired, but that is a matter for local agreement.

With this scheme, web origins can only "speak for themselves" so that if the backend is e.g. at port 8443 rather then the default 443, then the ZF MUST access https://backend.example.com:8443/.well-known/origin-scvb when considering modifications to the HTTPS/SVCB records for _8443._https.backend.example.com. As a consequence, a ZF will need to be configured to periodically refresh the JSON resource for each origin separately; that is, a ZF cannot "discover" from port 443 that some other service, for which the ZF is expected to update the DNS, is also present on another port on the same host.

5. The JSON structure for origin service binding info

```
{
  "regeninterval": 3600,
  "endpoints": [
    {
      "priority": 1,
      "params": {
        "ipv4hint": [ "192.0.2.1", "192.0.2.254" ],
        "ech": "AD7+DQA6NAAgACCn3zNTeX/WOD...AAA==",
        "ipv6hint": [ "2001:DB::ec4" ],
        "alpn": [ "h2", "http/1.1" ]
      }
    }
  ]
}
```

Figure 5: Sample JSON for ECH without aliases

```
{
   "regeninterval": 108000,
   "endpoints": [{
       "alias": "cdn1.example.com",
   }]
}
```

Figure 6: Sample JSON with aliasing

The JSON file at the well-known URI MUST contain an object with two keys: "regeninterval", whose value is a number, and "endpoints" whose value is an array of objects. All other keys MUST be ignored.

The "regeninterval" must be a positive integer and specifies the number of seconds between e.g. ECH key generation actions at the origin; that is, a replacement ECHConfigList may be generated this often. This is used to determine when to next poll the origin for updates, and MAY be used by the ZF to generate DNS TTL values, for example with a TTL that is half of the regeninterval.

More precise expiration times are common in other protocols (e.g., the "notAfter" field in certificate lifetime validity, discussed in Section 4.1.2.5 of [RFC5280]), but are too stringent for this use-case. The ZF cannot afford to fail open (by removing the HTTPS records) or fail closed (by removing the IP addresses), but it can safely "stretch" the lifetime of the HTTPS records because of ECH's "retry_configs" behavior. Since we have to accept this stretching, it makes sense to avoid an explicit expiration time and instead speak about the intended update frequency. This also make it clear the origin must tolerate some amount of version skew, and gives operational flexibility to avoid unreasonable update frequencies.

The "endpoints" key is an array of objects.

* An empty endpoints array MUST be treated as an error.

* An endpoints array with one element can be one of three things:

  1. An empty object, which the ZF should take to be an HTTPS record with inferred SvcPriority, a TargetName equal to ".", and no specific service binding data, e.g. no ECH support. This can can be useful as a simple way to make use of the HTTPS RR type's HSTS behavior.

  2. A ServiceMode entry, containing at least one key from the JSON HTTP Origin Info registry (see IANA Considerations (Section 9), below).

  3. An AliasMode entry that points to another DNS name that must be resolved.

* If the endpoints array has more than one element, every item SHOULD be a ServiceMode entry, due to restrictions on the use of multiple AliasMode records (see , Section 2.4.2 [RFC9460]).

This format is designed to allow full use of the capabilities of HTTPS records [RFC9460] in natural JSON while minimizing the risk of invalid configurations.

The following keys are defined for ServiceMode entries:

target: The value is a string containing a fully qualified domain name corresponding to the HTTPS record's TargetName with the final "." removed. The default value is the empty string (""), corresponding to a TargetName of ".". To avoid complex conversion logic, the characters in this value MUST be limited to lower ASCII letters, digits, "-", "_", and "." (ALPHA / DIGIT / "-" / "_" / "." in ABNF [RFC5234]). These are the characters used in preferred name syntax [RFC1034] and underscore labels.

priority: The value is a positive integer corresponding to the SvcPriority. If omitted, the ZF SHOULD use the SvcPriority from the previous element of the endpoints array (or the value one for the first element).

params: A JSON Dictionary representing the SVCB SvcParams. Each key

in the dictionary is a string containing a registered SvcParamKey name (e.g., "ipv6hint") or a SvcParamKey in generic form (e.g., "key65528"). Where a registered SvcParamKey name exists and is supported, the relevant generic form SHOULD also be supported, e.g. an implementation that supports "ech" SHOULD also accept "key5" as meaning the same thing. The default value is "{}".

* For single-valued SvcParams (e.g., "ech"), the value is a JSON String. A JSON String is a sequence of Unicode codepoints, while a SvcParam's presentation value is a sequence of octets, so each value octet is stored as a single Unicode codepoint [ISOMORPHIC-DECODE]. In almost all cases, this is equivalent to the ordinary string representation of the presentation value.

* For list-valued SvcParams (e.g., "alpn"), the value is a JSON Array of Strings. Each String represents an octet sequence, as in the single-value case.

The following key is defined for AliasMode entries.

alias: The value MUST be a DNS name that could be used as the TargetName of an HTTPS resource record. This indicates that the backend is hosted on the same endpoints as this target, and is equivalent to an HTTPS AliasMode record. The ZF might implement this directive by publishing an AliasMode record, publishing a CNAME record, copying HTTPS records from the target zone, or fetching https://cfs.example.com/.well-known/origin-svcb (if it exists).

These definitions, taken with the ZF behaviour (Section 6) specified below, provide the following important properties:

* Origins can express any useful configuration that is representable by HTTPS records, including multiple endpoints representing different ports, providers, etc.

* Origins that simply alias to a single target can indicate this without copying the ECHConfig and other parameters, avoiding the maintenance burden of staying synchronized with the target's key rotations and configuration updates.

If the origin makes use of intermediaries, it is the origin's responsibility to ensure that the origin-svcb JSON document correctly accounts for their current configuration.

Note that there are multiple, and possibly confusing, port numbers involved here. The port that is part of the web origin in the .well-known URL is part of the QNAME that will be used by clients in accessing the origin's HTTPS/SVCB resource records. The "port" field in the RR value, and in the JSON structure, is the value to be used in specifying an "alternative endpoint" as defined in section 1.3 [RFC9460].

6. Zone Factory behaviour

The behaviours of the zone factory can be divided into two categories: behaviors that occur once, when configuring the zone factory to use a JSON document, and synchronization behaviors that occur on an ongoing basis after configuration is complete.

6.1. Configuration behaviours

In its default configuration, the zone factory SHOULD NOT perform record synthesis from any origin-svcb JSON documents. Activating record synthesis SHOULD require a configuration change by the operator. This avoids surprising additions of records to the DNS zone.

A zone factory MAY offer a configuration option (off by default) to detect A and AAAA records in the DNS zone that correspond to HTTP origins with a well-known JSON resource, and automatically synthesize HTTPS records for all such origins. This procedure can work when the A or AAAA records' owner name is sufficient to generate the correct "https://" URL for the well-known JSON resource. Note that this mechanism is limited to "https://" origins with a port number of 443.

To generate specific HTTPS records, a zone factory MAY offer a configuration option to specify a list of secure HTTP origins. The zone factory can use these origins to produce well-known URIs (to retrieve the origin-svcb JSON resource) and port-prefixed owner names (for the generated HTTPS records).

To generate records with arbitrary SVCB-compatible record types, a zone factory MAY allow the operator to configure the full owner name and record type of the records to be generated, and the full URL or local filesystem path of an origin-svcb resource. A full URL is required because there is not, in general, an HTTP origin that is intrinsically authorized to control the content of a SVCB-compatible DNS record.

6.2. Ongoing synchronization behaviours

If the ZF is unable to convert the JSON into a DNS zone fragment (e.g., due to an unrecognized SvcParamKey), or if the resulting zone fails validation checks, the ZF MUST NOT update the DNS. Such failures will not be directly visible to the client-facing server, so ZF implementations will need to provide some form of reporting so that the situation can be resolved. Note that this can lead to inconsistent behavior for a single origin served by multiple ZFs.

A ZF MAY apply additional processing according to its own policy, such as adjusting TTL values and correcting common misconfigurations.

A ZF SHOULD check that ECH with the presented endpoints succeeds with the backend before publication. In order to make such checks, the ZF SHOULD attempt to access the well-known URI defined here while attempting ECH.

A bespoke TLS client is likely needed for this check, that does not require the ECHConfigList value to have already been published in the DNS. The TLS client also needs to allow checking for the success or failure of ECH.

Note that the ECHConfigList could contain some ECHConfig values that work and some that deliberately do not work if, for example, the TLS server wishes to GREASE it’s ECHConfigList according to [RFC9849] section 6.2.2.

If ipv4hints or ipv6hints are present, and if those are not the same values as are published in A/AAAA RRs for the backend, then the ZF SHOULD check that webPKI based authentication of the backend works at all of the relevant addresses.

A ZF SHOULD publish all the endpoints that are presented in the JSON file that pass the checks above.

A ZF SHOULD set a DNS TTL less than regeninterval, i.e. short enough so that any cached DNS resource records are likely to have expired before the JSON object’s content is likely to have changed. The ZF MUST attempt to refresh the JSON object and regenerate the zone before this time. This aims to ensure that ECHConfig values are not used longer than intended by backend.

7. Security Considerations

This document defines a way to publish SVCB/HTTPS RR values. If the wrong values were published in the DNS, then TLS clients using ECH might suffer a privacy leak, or degraded service due to overuse of ECH retry_configs.

Similarly, a ZF that also has write access to A/AAAA RRs for a backend, SHOULD NOT publish HTTPS RRs that contain ipv4hints or ipv6hints that are in conflict with the correct A/AAAA values unless those have been verified (via webPKI) as belonging to the same backend.

When considering the content of SVCB/HTTPS RRs, the general argument for the security of this scheme is that this scheme has the backend server authenticate the JSON structure that is mapped directly to the SVCB/HTTPS RR, to eventually be used by TLS clients when interacting with the backend server, via the client-facing server. Note that if an SVCB record is being published to support access via some non-HTTP protocol, then additional checks may be required.

If a ZF is also authoritative for the CAA RR [RFC8659] for a CFS, then, as part of verifying the origin of the .well-known JSON, that ZF is in a position to evaluate whether the CA that issues the CFS TLS server certificate was authorized to do so at the time of issuance. A check of this kind can reduce exposure to attacks by an untrustworthy CA.

ECH split-mode security also requires that the backend server acquire SvcParamKey values from the client-facing server via some authenticated means. If the backend server acquires the JSON data from the well-known URL and it is properly authenticated via HTTPS from the client-facing server’s public_name then that satisfies this requirement.

The system described here depends on the webPKI for authentication of entities and results in publication of new SVCB/HTTPS RRs. The webPKI itself, however, often depends on the DNS to demonstrate control over a DNS name, e.g. when using the ACME protocol [RFC8555] with the HTTP-01 challenge type. A temporary breach of a backend server that allows the attacker to contol the JSON content described here could be used to bootstrap more long-lasting control over the backend's DNS name if the attacker were to request new certificates during the time when the attacker's chosen values were published in the DNS, and if the ACME server doing the validation solely depended on content from the backend's HTTPS RR, e.g. preferring ipv6hints over the AAAA for the backend. It would seem prudent for ACME servers to be cautious if using ipv4hints and ipv6hints, e.g. flagging divergence between those values and A/AAAA RRs.

Although the .well-known URL defined here may well be publicly accessible, general HTTP clients SHOULD NOT attempt to use this resource in lieu of HTTPS records queries through their preferred DNS server for the following reasons:

* The bootstrap connection would not be able to use ECH, so it would reveal all the information that ECH seeks to protect.

* The origin could serve the user with a uniquely identifying configuration, potentially resulting in an unexpected tracking vector.

Operators should be careful to avoid directory enumeration or traversal attacks if the well-known URLs are served directly from a filesystem. For example, a misconfiguration of this kind could expose the ECH private keys.

In multi-CDN and similar scenarios, a ZF might only test ECH success against one of the CDNs unless the ZF can make use of the ipv4hints and/or ipv6hint values, or the ZF has out of band information about the different addresses at which backend.example.com can be accessed.

This document doesn't specify how origins are added to the list used by a ZF, nor how those are deleted, nor does it specify how a CFS might signal to a ZF that it wishes all HTTPS RRs to be deleted.

To reduce the frequency of DNS zone updates, ZF implementations MAY defer any DNS updates until they detect a a change to the JSON value. Changes can be detected using HTTP cache headers, comparison of normalized JSON or internal data structures to a cached copy, or comparison of the generated DNS records (including TTLs) with those currently published in the zone. If DNS records are retrieved for this purpose using a DNS query, it MUST use an authenticated secure

transport directly to the authoritative server, in order to ensure integrity of the returned TTL value. For clarity: a localhost connection between the ZF implementation and the authoritative server process that does not use the system stub resolver is considered sufficently secure for this protocol, should that be an acceptable local policy.

Using this protocol, a ZF SHOULD only publish RRs for origins that are actively taking part in this protocol. If an origin used to use this protocol, but has since stopped, and if a ZF was still polling for that origin's .well-known and the origin hadn't deleted the origin-svcb JSON, then undesired publication of RRs could occur. Implementations SHOULD take steps to try ensure such accidental publication does not happen.

8. Acknowledgements

Thanks to David Adrian, David Benjamin, David Blacka, Tim Chown, Ted Hardie, Watson Ladd, Niall O'Reilly, Yaakov Stein and Martin Thomson for reviews.

Stephen Farrell's work on this specification was supported in part by the Open Technology Fund.

9. IANA Considerations

IANA is requested to take two actions: registering a new well-known URI in the registry at https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml#well-known-uris-1 and creating a new registry for defining items in the JSON object found at that endpoint.

9.1. Well-known endpoint registration

IANA is requested to add the following entry to the Well-Known URIs table:

Column	Value
URI Suffix	origin-svcb
Change Controller	IETF
Reference	{This RFC}
Status	permanent
Related Information	Must be fetched via HTTPS
Date Registered	{When registered}
Date Modified	

Table 1: Additional Well-Known entry

Items in curly braces should be replaced with their actual values.

9.2. JSON Service Binding Info

If approved, this specification requests the creation of an IANA registry named "JSON Service Binding Info" with a Standards Action registration policy. The request is to put the table in a new file "json-svcb.xml" in the existing "dns-svcb" registry group. The table has three columns:

Name: the name of the top-level field being added

Reference: the document that defines the semantics of the field

Notes: any short additional information the registrant wishes to add

The table should be populated with the following two entries, where Items in curly braces should be replaced with their actual values, and the "Notes" column is empty.

Name	Reference	Notes
endpoints	{This RFC}	
regeninterval	{This RFC}	

Table 2: Initial values for the registry

If additional entries are added to this table, then the relevant RFC should update this one.

10. References

10.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[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>.

[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/info/rfc5234>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[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>.

[RFC8446] 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>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC9460] Schwartz, B., Bishop, M., and E. Nygren, "Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)", RFC 9460, DOI 10.17487/RFC9460, November 2023, <https://www.rfc-editor.org/info/rfc9460>.

[RFC9848] Schwartz, B., Bishop, M., and E. Nygren, "Bootstrapping TLS Encrypted ClientHello with DNS Service Bindings", RFC 9848, DOI 10.17487/RFC9848, March 2026, <https://www.rfc-editor.org/info/rfc9848>.

[RFC9849] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", RFC 9849, DOI 10.17487/RFC9849, March 2026, <https://www.rfc-editor.org/info/rfc9849>.

[I-D.ietf-tls-key-share-prediction] Benjamin, D., "TLS Key Share Prediction", Work in Progress, Internet-Draft, draft-ietf-tls-key-share-prediction-04, 19 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-key-share-prediction-04>.

10.2. Informative References

[RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019, <https://www.rfc-editor.org/info/rfc8555>.

[RFC8659] Hallam-Baker, P., Stradling, R., and J. Hoffman-Andrews, "DNS Certification Authority Authorization (CAA) Resource Record", RFC 8659, DOI 10.17487/RFC8659, November 2019, <https://www.rfc-editor.org/info/rfc8659>.

[ISOMORPHIC-DECODE] WHATWG, "WHATWG definition of Isomorphic Decode", <https://infra.spec.whatwg.org/#isomorphic-decode>.

Appendix A. Change Log

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

The -00 WG draft replaces draft-farrell-tls-wkesni-03.

Version 01 changed from a special-purpose design, carrying only ECHConfigs and port numbers, to a more general approach based on Service Bindings.

Version 02 is just a keep-alive

Version 03 reflects some local implementation experience with -02

Version 04 matches a proof-of-concept bash script implementation and results of IETF-117 discussion.

Version 05 updated the IANA and Security considerations and fixed conformance to HTTPS SVCB spec. It also addressed early artart and dnsop reviews, and some list discussion/github issues.

Version 06 changed the name and some of the text because it is no longer ECH-specific.

Version 07 clarifies that origins only speak for themselves and has editorial changes and clarifications.

Version 08 resolves a number of issues identified in discussion between authors and arising from implementation and (a very tiny) deployment.

Version 09 generalised the text some more in response to comments at IETF-123 and resolved a few more issues and updated the acks section.

Versions 10 to 12 are mostly keep-alives and updating refs.

Authors' Addresses

Stephen Farrell
Trinity College Dublin
Dublin
2
Ireland
Phone: +353-1-896-2354
Email: stephen.farrell@cs.tcd.ie

Rich Salz
Akamai Technologies
Email: rsalz@akamai.com

Benjamin Schwartz
Meta Platforms, Inc.

Email: ietf@bemasc.net