Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 22, 2018

P. Hallam-Baker
Comodo Group Inc.

September 18, 2017

JSON Web Service Binding Version 1.0
draft-hallambaker-json-web-service-07

Abstract

The JSON Web Binding (JWB) describes a standardized approach to implementing Web Services. Services are advertised using the DNS SRV and HTTP Well Known Service conventions. Messages may be authenticated or authenticated and encrypted at the message layer in addition to any transport and/or network layer security. Service messages are encoded in JSON using Internet time format for Date-Time fields and Base64urlencoding for binary objects.

This document specifies Version 1.0 of JWB which uses HTTP/1.1 for transport. Use of the multiple stream capabilities of HTTP/2 or QUIC is outside the scope of this document.

This document is also available online at http://prismproof.org/Documents/draft-hallambaker-json-web-service.html [1] .

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 March 22, 2018.

Table of Contents

1. Introduction

The JSON Web Binding (JWB) specifies one approach to using DNS Discovery [RFC1035] [RFC1035] , the HTTP [RFC7230] [RFC7230] protocol and the JSON data encoding [RFC7159] [RFC7159] in a Web Service.

JWB is not the only approach possible, but developing a standard means making choices that don?t matter to developers that build on it. While there are infinitely many ways that a Web Service might employ HTTP and JSON to implement a Web Service, a client and a server can only interoperate if they both agree to use the same one.

Beginning with a DNS address of the service (e.g. example.com), the client identifies a specific HTTP URL at which to access the service. The DNS SRV record [RFC2782] [RFC2782] and Well Known Service [RFC5785] [RFC5785] mechanisms are used for this purpose.

Web Services MAY require authentication and encryption services at the message level even if transport layer security (e.g. TLS [RFC5264] [RFC5264]) is used. Use of such enhancements is signaled using the HTTP Content-Encoding header.

The mapping of data types described in the specification (integer, string, etc.) to JSON data types. [RFC3339] [RFC3339] Date time encoding and BASE64-url encoding [RFC4648] [RFC4648] are used to map date-time and binary data types to JSON encoding values.

JWB establishes an intermediate layer between the Web Service and the network layer (Figure 1).

[[This figure is not viewable in this format. The figure is available at http://prismproof.org/Documents/draft-hallambaker-json-web-service.html [2].]]

JWB Defines Presentation and Encoding Layers.

A key architectural principle that guides the design of JWB is that the Web Service implementation should be as independent of the HTTP presentation layer as is possible. Thus:

Message semantics are not affected by HTTP headers or the request line URL.

The use of HTTP response codes is limited to reporting errors and warnings that arise from the HTTP transport.

If message layer authentication or authenticated encryption are used, this is applied within the HTTP content payload and not through a combination of payload and header data.

This specification describes a mechanism for accessing a collection of hosts as a single undifferentiated service with provision for load balancing and fault tolerance. This has important consequences for state management. Web Services typically involve some form of stateful interaction or real world side effect. Otherwise, it is likely that the Web Service would be better written as a traditional Web interaction mapping the stateless resources to URIs.

The mechanism described in this specification is intended for the initial discovery of a host with which to engage in a Web Service transaction which may or may not consist of a series of message exchanges. Since sharing state between hosts supporting a virtual service requires resources and typically introduces latency, a service specification MAY require that a transaction begun on one host be completed with the same host and the time period in which a transaction that is accepted by one host will be regarded as ?final? by the virtual service.

For example, a file upload protocol for a photograph sharing service might have separate messages for checking that there is space to store the photograph ?Check?, uploading the file ?Store? and reporting that the data is safely stored at multiple locations. When responding to the Check command, the host is reporting that there is space at that local node. It is obviously undesirable for a client to verify that host1 has enough space to store the file and then attempt to upload the file to host2. Equally, having uploaded the file to host1, it might be minutes, hours or even days before the

photograph could be retrieved through host2. Such questions are left for the Web Service protocol designer to address.

2. Definitions

This section presents the related specifications and standard, the terms that are used as terms of art within the documents and the terms used as requirements language.

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.2. Defined Terms

No terms of art are defined.

2.3. Related Specifications

TBS!!!!

2.4. Implementation Status

The implementation status of the reference code base is described in the companion document [draft-hallambaker-mesh-developer] [draft-hallambaker-mesh-developer] .

3. Service Discovery

Service discovery is the process of resolving the address of a Web Service to a Web Service Endpoint, a URI [RFC3986] [RFC3986] at which the service is provided.

A JWB Web Service is specified by giving the DNS Address of the service <domain>, and Well Known Service name <service>.

3.1. Host Identification

The first step in service discovery is to resolve the <domain> and <service> identifiers to the IP address of a host that provides that service.

3.1.1. SRV Host discovery

A client attempting to connect to the service first attempts to locate an SRV record [RFC2782] [RFC2782] for the specified service:

```
_<service>._tcp._<domain>  SRV  <priority> <weight> <port> <host>
```

Figure 1

Where <priority> and <weight> are the SRV priority and weight parameters specified in [RFC2782] [RFC2782] , <port> is the TCP port number and <host> is the DNS name of the host for which the service advertisement is made. Standard A/AAAA/CNAME resolution is used to obtain the IP address of the host from <mapping>.

If a match is found, the client uses the mechanism specified in [RFC2782] [RFC2782] to choose hosts and attempts to contact each host in turn until a successful HTTP connection is established or the maximum number of attempts threshold is reached.

3.1.2. DNS Fallback

Despite the fact that SRV records have been a part of the DNS standard for 20 years, it is not uncommon for network intermediaries to implement SRV record resolution incorrectly or block it entirely.

If no SRV record is found, a client MAY perform fallback discovery if explicitly authorized to do so by the corresponding Web Service protocol specification. The client attempts to connect to the host <service>.<domain> using the standard A/AAAA/CNAME resolution rules to obtain the IP address of the host.

3.1.3. TXT Service Description

A service MAY advertise service and/or host description information using TXT records. These have the following format

```
_<service>._tcp._<domain>  TXT "<tag>=<value> [<tag>=<value>]*"
_<service>._tcp._<host>  TXT "<tag>=<value> [<tag>=<value>]*"
```

Figure 2

Service descriptions specified under the domain address of the service apply to all host instances of the service. Descriptions specified under the domain address of a host instance apply only to that host instance and take precedence over values specified at the service level.

The following tags are defined:

The path to use to construct the Web Service Endpoint.

The service version(s) supported in the format <max>-<min>

3.2. HTTP host processing

Having identified the IP address, the client performs a HTTP or HTTPS Web Service POST request at the default Web Service Endpoint specified by the Well Known Service name and the DNS address of the host instance.

http://<host>:<port>/.well-known/<service>

Figure 3

Note that a given host MAY provide multiple instances of a Web Service under different discovery addresses. Therefore it is essential to use the original <domain> value rather than the <mapping> returned in the SRV record.

3.2.1. Use of TLS transport

This document does not describe a mechanism for mandating use of TLS.

If TLS is used, the Web Service client MUST use the service address (<domain>) as the basis for certificate subjectAltName validation.

3.2.2. Fallback Processing Rules

If a client?s attempt to connect to a host selected from an SRV connection redirection results in a HTTP (3xx), Client error (4xx) or Server error (5xx) code, the client MUST process the HTTP error response and not simply attempt a connection to a different host. If a client request is rejected for lack of authentication (511) or because the request is too large (413) at one host, the client should assume that the request will be rejected for the same reason at another. If the attempt to create a TCP connection fails or the server returns Service Unavailable (503), the client MAY use the SRV fallback rules to select an alternative host.

3.2.3. Service Continuation

Once the initial service discovery mechanism has been used to establish contact with a host, the service protocol MAY specify that further interactions be directed to another host and/or using a

different protocol. Such mechanisms are outside the scope of this specification.

3.3. Example

The Mathematical Mesh has the Well Known Service name of ?MMM'. Accounts used in the Mathematical Mesh follow the [RFC5322] [RFC5322] format of <user>@<domain>.

Alice has the account alice@example.com and the DNS configuration file for example.com has the following entries:

```
_mmm._tcp.example.com SRV host1.example.com 0 10 80 host1.example.com
_mmm._tcp.example.com SRV host2.example.com 0 40 80 host2.example.com
_mmm._tcp.example.com TXT "version=1.0-2.0"
mmm.example.com       CNAME host3.example.com
host1.example.com     A 10.0.1.1
host2.example.com     A 10.0.1.2
_mmm._tcp.host2.example.com TXT "path=/service"
host3.example.com     A 10.0.1.1
host3.example.com     A 10.0.1.2
```

Figure 4

The client attempts to resolve the address alice@example.com as follows:

1. Client attempts to resolve SRV and TXT records for _mmm._tcp.example.com

2. DNS resolver returns two SRV entries and one TXT entry

3. Client makes a random selection between host1 (20% weighting) and host2 (80% weighting). Chooses host1.

4. Client resolves A/AAAA for host1.example.com and TXT for _mmm._tcp.host1.example.com

5. DNS resolver returns A=10.0.1.1 and TXT=none

6. Client attempts to POST Web Service request to http://host1example.com/.well-known/mmm at host address 10.0.1.1

7. The host at 10.0.1.1 returns 503 Service Unavailable

8. Client resolves A/AAAA for host2.example.com and TXT for _mmm._tcp.host2.example.com

9. DNS resolver returns A=10.0.1.2 and TXT "path=/service"

10. Client attempts to POST Web Service request to http://host2example.com/service at host address 10.0.1.2

11. Request succeeds, session proceeds.

If the same client is used in a network location where the SRV record resolution fails due to a faulty firewall configuration, the resolution proceeds as follows:

1. Client attempts to resolve SRV record for _mmm._tcp.example.com

2. DNS resolver returns ?not found?

3. Client attempts to resolve A and AAAA record

4. DNS resolver returns 10.0.1.1, 10.0.1.2

5. Client makes a random selection between 10.0.1.1 (50% weighting) and 10.0.1.2 (50% weighting). Chooses host1.

6. Client attempts to POST Web Service request to http://example.com/.well-known/mmm at host address 10.0.1.1

7. The host at 10.0.1.1 returns 503 Service Unavailable

8. Client attempts to POST Web Service request to http://example.com/.well-known/mmm at host address 10.0.1.2

9. Request succeeds, session proceeds.

Note that the main differences between these two scenarios is that the use of the SRV record allows the service configuration to account for load balancing with tiers of fallback support and use of service description information while the use of round robin A/AAAA records does not.

4. HTTP Messages

JWB messages are exchanged as HTTP POST transactions. Support for and use of HTTP/1.1 [RFC7230] [RFC7230] is REQUIRED unless otherwise specified by the Web Service Specification.

While the use of HTTP/2 [RFC7540] [RFC7540] offers the potential benefit of multiple concurrent transaction streams, the means of making use of such capabilities is outside the scope of JWB v1.0 but is likely to be the main incentive for defining a revision.

In contrast to other approaches to the design of Web Services, the only use made of the HTTP transport is to distinguish between different services on the same host using the Host header and .well-known convention and for message framing.

No use is made of the URI request line to identify commands, nor are the caching or proxy capabilities of HTTP relied on. One of the main design objectives of JWB is to enable message level authentication. Since HTTP headers are mutable and may be changed by intermediaries, any attempt to make use of HTTP features requires a mechanism to canonicalize or duplicate the headers. Furthermore, the implementation of authentication and encryption features at the message level is liable to be incompatible with the HTTP caching model and any attempt to implement caching is moot when TLS is in use.

4.1. Request

The HTTP request MAY contain any valid HTTP header specified in [RFC7230] [RFC7230] .

/well-known/<service> (unless overridden using a TXT path attribute)

POST

<domain>

As specified in section yy below.

As specified in section zz below.

As specified in [RFC7230] [RFC7230] .

The content payload as specified in section XX below.

Example: The HTTP request for the mmm service in the previous example would be:

```
POST /.well-known/mmm HTTP/1.1
Host: example.com
Content-Type: application/json
Content-Length: 16

{ ?hello? : {} }
```

Figure 5

4.2. Response

The response MAY contain any HTTP response header but since JWB services do not make use of HTTP caching and messages are not intended to be modified by HTTP intermediaries, only a limited number of headers have significance:

The HTTP response code. This is processed as described in section zz below.

As specified in section zz below.

As specified in [RFC7230] [RFC7230] .

Since the only valid HTTP method for a JWB request is POST, JWB responses are not cacheable. The use of the cache-control header is therefore unnecessary. However, experience suggests that reviewers find it easier to understand protocol specifications if they are reminded of the fact that caching is neither supported nor desired.

Example: The HTTP response for the mmm service in the previous example would be:

```
HTTP/1.1 200 OK
Content-Type: application/json
Connection: keep-alive
Cache-Control: no-store
Content-Length: 43

{ ?hello-response? : { ?Version? : ?1.0? }}
```

Figure 6

5. Error handling and response codes

A JWB Web Service is effectively using a three layer protocol stack with the potential for an error to occur at any of the three layers:

Transport Layer

HTTP Layer

Web Service Layer

Services SHOULD always attempt to return error codes at the highest level possible. However, it is clearly impossible for a connection that is refused at the Transport layer to return an error code at the

HTTP layer. It is however possible for a HTTP layer error response to contain a content body.

In the case that a JWB response contains both a HTTP response code and a well formed JWB payload containing a response, the JWB payload response SHALL have precedence.

6. Content Encoding

The HTTP Content-Encoding header specifies transformations performed on the content before the HTTP Transfer encoding was applied. This is commonly used for specifying compression. In JWB the Content-Encoding header MAY be used to specify that the content that follows contains a payload that is either authenticated [RFC7515] [RFC7515] or authenticated and encrypted [RFC7516] [RFC7516] using the JOSE specification.

6.1. Direct Encoding

If the Content-Encoding header is absent or empty, the HTTP content is the message payload as specified by the Content-Type header.

6.2. Content-Encoding: jose-jwb

The Content-Encoding type jose-jwb is a serialization format for JSON Web Signature and JSON Web Encryption objects. Each message consists of the following sequence:

Preamble: A JSON object in UTF-8 encoding

ASCII Record Separator Character (%x1E)

Payload

ASCII Record Separator Character (%x1E)

Postscript: A JSON object in UTF-8 encoding

The payload data consists of all the data that appears between the first and the last occurrence of the record separator character %x1E in the HTTP content. Since the UTF-8 encoding does not permit the octet %0x1E to occur within a well formed JSON object, the use of this character as a record separator is unambiguous even if the character occurs within the payload (as is possible with a binary content-type or if the payload is encrypted).

The contents of the Preamble, Payload and Postscript vary according to whether the message is authenticated or authenticated and encrypted.

6.3. Authenticated

Authenticated messages are signed using Jose Web Signature [RFC7515] [RFC7515] . The Preamble, Payload and Postscript are formed as follows:

A JSON Object containing the JWS Protected Header

The binary data over which the signature value is calculated

The JWS Signature header

Note that a jose-jwb message is only permitted to have a single header and there is no provision for providing data that is not integrity protected.

6.4. Authenticated Encryption

Encrypted messages are signed using Jose Web Signature [RFC7516] [RFC7516] . The Preamble, Payload and Postscript are formed as follows:

A JSON Object containing the JWS Protected Header

The binary data over which the signature value is calculated

The JWS Encryption header

Note that a jose-jwb message is only permitted to have a single header and there is no provision for providing data that is not integrity protected.

7. Content Type

The Content Type header specifies the plaintext payload media type as specified in [RFC6838] [RFC6838] .

For version1.0 of this specification, the only supported payload media type is application/json as specified in [RFC7159] [RFC7159] .

Future versions of this specification may include support for binary encodings such as JSON-B [draft-hallambaker-jsonbcd] [draft-hallambaker-jsonbcd] and/or CBOR [RFC7049] [RFC7049] .

8. JSON Data Bindings

Note that this is the only part of this specification that is strictly limited to JSON encoding. The rest of the specification is equally applicable to Web Services using XML and/or CBOR encoding.

8.1. Request Message

Each JWBv1.0 request contains exactly one Web Service Command. [Future versions MAY specify a mechanism that permits multiple commands to be sent in parallel]

The request payload contains a JSON object that contains exactly one member whose name is the name of the command that is requested and whose value is an object that contains the command parameters (if any).

```
{ ?hello? : {} }
```

Figure 7

8.2. Response Message

The response payload contains a JSON object that contains the members specified by the Web Service specification.

Future versions of this specification MAY reserve particular fields in the response payload for particular purposes (e.g. returning status values).

```
{ ?hello-response? : { ?Version? : ?1.0? }}
```

Figure 8

8.3. Data Fields

JSON was originally developed to provide a serialization format for the JavaScript programming language [ECMA-262] [ECMA-262] . While this approach is generally applicable to the type systems of scripting programming languages, it is less well matched to the richer type systems of modern object oriented programming languages such as Java and C#.

Working within a subset of the capabilities of JSON allows a Web Service protocol to be accessed with equal ease from either platform type. The following capabilities of JSON are avoided:

The ability to use arbitrary strings as field names.

The use of JSON objects to define maps directly

The following data field types are used:

Integer values are encoded as JSON number values.

Test strings are encoded as JSON text strings.

Boolean values are encoded as JSON ?false?, ?true? or ?null? tokens according to value.

Sequences of data items that are encoded as JSON arrays

Objects whose type is known to the receiver are encoded as JSON objects

Objects whose type is not known to the receiver are encoded as JSON objects containing a single field whose name describes the type of the object value and whose value contains the value.

Byte sequences are converted to BASE64-url encoding [RFC4648] [RFC4648] and encoded as JSON string values.

Date Time values are converted to Internet time format as described in [RFC3339] [RFC3339] and encoded as JSON string values.

9. Security Considerations

A fuller treatment of the security considerations will follow.

9.1. Integrity

9.1.1. DNS Spoofing

9.1.2. TLS Downgrade

9.1.3. TLS Service Impersonation

9.1.4. Request Replay Attack

9.1.5. Response Replay Attack

9.2. Confidentiality

9.2.1. Side Channel Attack

9.2.2. Session Key Leakage

10. IANA Considerations

The following registrations are required:

jose-jwb

/.well-known/srv/

[Or change registry to FCFS]

11. References

11.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006.

[RFC5264] Niemi, A., Lonnfors, M., and E. Leppanen, "Publication of Partial Presence Information", RFC 5264, DOI 10.17487/RFC5264, September 2008.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015.

11.2. Informative References

[draft-hallambaker-jsonbcd]
Hallam-Baker, P., "Binary Encodings for JavaScript Object Notation: JSON-B, JSON-C, JSON-D", draft-hallambaker-jsonbcd-07 (work in progress), August 2017.

[draft-hallambaker-mesh-developer]
Hallam-Baker, P., "Mathematical Mesh: Reference Implementation", draft-hallambaker-mesh-developer-03 (work in progress), August 2017.

[ECMA-262]
Ecma International, "ECMAScript? 2017 Language Specification", June 2017.

[RFC5322] Resnick, P., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015.

11.3. URIs

[1] http://prismproof.org/Documents/draft-hallambaker-json-web-service.html

[2] http://prismproof.org/Documents/draft-hallambaker-json-web-service.html

Author's Address

Phillip Hallam-Baker
Comodo Group Inc.

Email: philliph@comodo.com