Internet Engineering Task Force (IETF)
Internet-Draft
Intended Status: Standards Track
Expires: November 20, 2014

Phillip Hallam-Baker
Comodo Group Inc.

May 19, 2014

OmniBroker Protocol
draft-hallambaker-omnibroker-08

Abstract

An Omnibroker is an agent chosen and trusted by an Internet user to provide information such as name and certificate status information that are in general trusted even if they are not trustworthy. Rather than acting as a mere conduit for information provided by existing services, an Omnibroker is responsible for curating those sources to protect the user.

The Omnibroker Protocol (OBP) provides an aggregated interface to trusted Internet services including DNS, OCSP and various forms of authentication service. Multiple transport bindings are supported to permit efficient access in virtually every common deployment scenario and ensure access in any deployment scenario in which access is not being purposely denied.

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 http://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."

Copyright Notice

Table of Contents

1. Definitions 4
1.1. Requirements Language 4
2. Purpose 4
2.1. Omnibroker Discovery and Publication Services 4
2.2. Omnibroker Implementation 4
2.2.1. Establishing service 5
2.2.2. Protocol Bindings 5
3. OmniDiscovery Service 6
3.1. Related Work 6
4. Walled Gardens 6
4.1. Censorship 7
4.2. Trust Substitution 7
4.3. Censorship Bypass 8
5. Use 8
5.1. Connection Broker 8
5.1.1. Service Connection Broker 8
5.1.2. Peer Connection Broker 10
5.1.3. Credential Validation 11
5.1.4. Message: QMessage 12
5.1.5. Message: QRequest 12
5.1.6. Message: QResponse 12
5.1.7. Structure: Identifier 12
5.1.8. Structure: Connection 13
5.1.9. Structure: Credential 13
5.1.10. Structure: CertificateID 13
5.1.11. Structure: Advice 13
5.1.12. Structure: Service 14
6. OBPQuery 14
6.1. QueryConnect 14
6.1.1. Message: QueryConnectRequest 14
6.1.2. Message: QueryConnectResponse 14
6.2. Validate 15
6.2.1. Message: ValidateRequest 15
6.2.2. Message: ValidateResponse 15
7. Transport Bindings 15
7.1. JSON Payload Binding 16
8. Acknowledgements 17
9. Security Considerations 17
9.1. Denial of Service 17
9.2. Breach of Trust 17
9.3. Coercion 17
10. IANA Considerations 17
11. Example Data 17
11.1. Ticket A 17
11.2. Ticket B 17
12. References 17
12.1. Normative References 17
Author’s Address 18

1. Definitions

1.1. 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. Purpose

Today, a network client is required to make queries against multiple information sources to establish a secure connection to a network resource. A DNS query is required to translate network names to Internet addresses. If TLS transport is used, an OSCP query may be required to validate the server certificate. Support for client authentication may require interaction with another service.

Servers require similar support when accepting Internet connections. Even though most networking infrastructure supports some form of network administration, it is left to the network administrator to fill in the gap between server applications and network infrastructure. Making use of such facilities is rarely cost effective except at the very largest installations.

An Omnibroker is a trusted agent that acts as a single point of service for client queries requesting a connection to a named network resource and server advertisements accepting connections to a named network resource.

2.1. Omnibroker Discovery and Publication Services

The Omnibroker protocol is a meta-directory access protocol. As with any directory protocol, the two principal functions supported by Omnibroker are discovery and publication. These functions are supported by the OmniDiscover and OmniPublish Web Services.

This specification document describes the architectural approach shared by both protocols and the OmniDiscover protocol. The OmniPublish protocol is described separately in [I-D.hallambaker-omnipublish].

2.2. Omnibroker Implementation

Omnibroker Discovery and Omnibroker Publication make use of the following mechanisms defined in other specifications:

Service Connection Service (SXS) [!I-D.hallambaker-wsconnect]
: To establish and manage the long term trust relationship with the Omnibroker provider.

HTTP Session Authentication [!I-D.hallambaker-httpsession]
To provide message authentication in the HTTP/REST transport.

UDP Framed Messaged (UYFM) described in [!I-D.hallambaker-wsconnect]
For low latency transactions.

JSON Encoding [!RFC4627]
For encoding messages in the HTTP transport.

JSON Binary and Compressed encodings described in [!I-D.hallambaker-jsonbcd]
For efficient encoding messages in the low latency UYFM transport.

2.2.1. Establishing service

In normal use, an omnibroker client receives service from a single Omnibroker service provider. For performance and reliability reasons, an Omnibroker service provider is expected to provide multiple Omnibroker service instances.

An Omnibroker client acquires the network address information and credentials necessary to access an omnibroker service using the JCX Web Service to establish a connection binding. To ensure reliabilty and the ability to access the service in all circumstances, an Omnibroker connection binding SHOULD specify multiple service instances.

2.2.2. Protocol Bindings

Due to the need for low latency and the need to function in a compromised network environment, two protocol bindings are defined:

* A HTTP binding using HTTP [!RFC2616] for session layer framing and HTTP Session Continuation [!I-D.hallambaker-httpsession] for message authentication and JSON encoding [!RFC4627] of protocol messages.

* A UDP Binding using UYFM framing [!I-D.hallambaker-wsconnect] and JSON-B encoding [!I-D.hallambaker-jsonbcd] for framing and encoding of protocol messages.

The implementation overhead of support for three different protocol bindings is reduced by the choice of a binary encoding for JSON (JSON-B) that is very close in structure to JSON encoding allowing encoders and decoders to support both encodings with minimal additional code.

Regardless of the protocol binding used, all Omnibroker messages are authenticated with protection against replay attack under the cryptographic credentials established in the connection binding service instance.

3. OmniDiscovery Service

Directing queries through a single point of contact has performance, relability and security advantages. Directing queries to multiple network information sources degrades performance and may cause a connection request to fail if an information resource is not available. This has led many application providers to limit the information sources they consult. Directing queries through an Omnibroker allows as many information sources to be brought to bear as the broker has local cached data for without loss of performance or reliability.

Making use of additional data sources allows the broker to ’curate’ the response. If the broker knows that a Web site always returns a redirect to a TLS secured version of the same site, it can tell a Web Browser to go straight to the secure version. If a Web Server is hosted on a known botnet, the Omnibroker can tell the client that it really does not want to visit that location.

Unlike the traditional DNS configuration, an Omnibroker client decides which source(s) of trusted information to use rather than relying on whatever happens to be the nearest source to hand.

The traditional DNS approach creates an obvious security risk as DNS is a trusted service and deciding to choose a random DNS service advertised by the local DHCP service is clearly a poor decision process for a trusted service. Further the DNS protocol does not protect the confidentiality or integrity of messages exchanged.

3.1. Related Work

Omnibroker provides security for interactions with a DNS service by replacing the DNS protocol with a new protocol that provides a higher level abstract service. [I-D.hallambaker-privatedns] applies the same approach and platforms to provide confidentiality and integrity for legacy DNS protocol messages.

4. Walled Gardens

IETF culture has traditionally resisted attempts to establish partitions within the open Internet with restricted access to network resources or compromised security. Such ’Walled Gardens’ models typically exist for the benefits of those who own the walls rather than those forced to live inside them.

While virtually all residential Internet users reject such controls, most find them acceptable, if not desirable in workplaces and schools.

Omnibroker simplifies the process of establishing such a walled garden but does not make the walls any easier to defend.

4.1. Censorship

From a censorship point of view, the censorship concerns of running an Omnibroker are essentially the same as those of running a DNS service. The party who decides which discovery service to use can determine which content is visible to the users.

4.2. Trust Substitution

Like SCVP [RFC5055] /> and XKMS [TBS], Omnibroker permits an Internet client to delegate some or all aspects of PKIX [RFC5280] certificate path chain discovery and validation.

In the normal mode of operation, the Omnibroker service performs only path chain discovery, leaving the client to re-check the PKIX certificate path before relying on it. This gives the Omnibroker the power to veto a client connection to a server that it considers to be unsafe but not the power to tell the client to trust a site of its own choosing.

This ability to veto but not assert trust is appropriate and sufficient for the vast majority of network applications. It allows the broker to make use of additional path validation checks that are not supported in the client such as DANE [RFC6698] or Certificate Transparency [RFC6962] />.

There are however some workplace environments where the ability to access external network resources with strong encryption is not permissible by enterprise policy or in some cases by law. An intelligence analyst working at the NSA may have a need to access external Web sites that contain important information but must on no account have access to a covert channel that could be used to exfiltrate information. Certain Financial institutions with access to valuable commercial information are required to monitor and record all communications into and out of the company to deter insider trading.

The traditional response to such needs has been to tell the parties affected to look elsewhere for support. As a consequence the techniques used to satisfy such requirements are generally unfriendly to network applications in general and have in some cases put the public Web PKI trust infratructure at risk.

There is an argument to be made that rather than attempting to prohibit such activities entirely, it would be better to provide a principled method of achieving those ends and for mainstream software providers to support it in such a fashion that ensures that network applications configured for that mode of use can be readilly identified as such by end users.

4.3. Censorship Bypass

As the preceeding examples demonstrate, a party with control over the Omnibroker service chosen by a user has full control over the network activities of that user. An important corrolary of this fact is that all a user need do to achieve full control over their network activities is to run their own Omnibroker service and connect to that.

For example such an Omnibroker service might be configured to return connection data for permitted domestic Web sites as normal but direct attempts to connect to forbidden foreign news or social media through a privacy network such as TOR.

5. Use

For illustrative purposes, all the examples in this section are shown using the Web Services Transport binding. The security connection has already been established as described in [I-D.hallambaker-wsconnect].

5.1. Connection Broker

The OBP service connection broker answers the query ’what connection parameters should be used to establish the best connection to interract with party X according to protocol Y. Where ’best’ is determined by the Omnibroker which MAY take into account parameters specified by the relying party.

5.1.1. Service Connection Broker

The OBP service connection broker supports and extends the traditional DNS resolution service that resolves a DNS name (e.g. www.example.com) to return an IP address (e.g. 10.1.2.3).

When using an Omnibroker as a service connection broker, a client specifies both the DNS name (e.g. www.example.com) and the Internet protocol to be used (e.g. _http._tcp). The returned connection parameters MAY include:

The IP protocol version, address and port number to establish a connection to. If appropriate, a security transport such as TLS or IPSEC. If appropriate, a description of a service credential such as a TLS certificate or a constraint on the type of certificates that the client should consider acceptable. If appropriate, application

protocol details such as version and protocol options.

If an attempt to connect with the parameters specified fails, a client MAY report the failure and request a new set of parameters.

5.1.1.1. Service Connection Broker Example

Alice uses her Web browser to access the URL http://www.example.com/. The Web browser sends a QueryConnectRequest request to obtain the best connection parameters for the http protocol at www.example.com:

```
POST /.well-known/omni-query/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Session: Value=5_AlS4yMTeE82T6ZP9qAZN7TOhXtvqZ__zsLOmCxNrQ;
  Id=o7znkpTHfrqcwsI1eHkPghCj7YsGUCp0KV2DcV1qXGlCt9wzmr2T6UcO_0YI
  AcEqVdTsqRsYBtVNGs9SJyTCnMvjIlU1xQ9ZzoUtqtJsT4A
Host: localhost:8080
Content-Length: 123
Expect: 100-continue

{
  "QueryConnectRequest": {
    "Identifier": {
      "Name": "Example.com",
      "Service": "_http",
      "Port": 80}}}
```

The service responds with an ordered list of possible connections. In this case the site is accessible via plain TCP transport or with TLS. Since TLS is the preferred protocol, that connection is listed first.

```
   HTTP/1.1 OK Success
   Content-Length: 371
   Date: Mon, 19 May 2014 17:17:43 GMT
   Server: Microsoft-HTTPAPI/2.0

   {
     "QueryConnectResponse": {
       "Status": 200,
       "StatusDescription": "Success",
       "Connection": [{
           "IPAddress": "10.3.2.1",
           "IPPort": 443,
           "Transport": "TLS",
           "TransportPolicy": "TLS=Optional",
           "ProtocolPolicy": "Strict"},
         {
           "IPAddress": "10.3.2.1",
           "IPPort": 80,
           "ProtocolPolicy": "Strict"}]}}
```

5.1.2. Peer Connection Broker

Each OBP request identifies both the account under which the request is made and the device from which it is made. An OBP broker is thus capable of acting as a peer connection broker service or providing a gateway to such a service.

When using Omnibroker as a peer connection broker, a client specifies the account name and DNS name of the party with which a connection is to be established (e.g. alice@example.com) and the connection protocol to be used (e.g. _xmpp-client._tcp)

The returned connection parameters are similar to those returned in response to a service broker query.

5.1.2.1. Service Connection Broker Example

Although the QueryConnectResponse returned the hash of a PKIX certificate considered valid for that connection, the server returns a different certificate which the client verifies using the ValidateRequest query.

```
[POST /.well-known/omni-query/ HTTP/1.1
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Session: Value=ebgTLvWjZFeTMnowmomslUu9rPvzAPAciO11QK26NQg;
  Id=o7znkpTHfrqcwsI1eHkPghCj7YsGUCp0KV2DcV1qXGlCt9wzmr2T6UcO_0YI
  AcEqVdTsqRsYBtVNGs9SJyTCnMvjIlU1xQ9ZzoUtqtJsT4A
Host: localhost:8080
Content-Length: 1126
Expect: 100-continue

{
  "ValidateRequest": {
    "Service": {
      "Identifier": [{
          "Name": "example.com"}]},
    "Credential": [{
        "Data": "
MIIC0DCCAbigAwIBAgIQQut6m1F0PodIjIzop_d1uDANBgkqhkiG9w0BAQUFADAR
MQ8wDQYDVQQDEwZWb29kb28wHhcNMTMwNjI2MTczOTQyWhcNMTQwNjI2MDAwMDAw
WjARMQ8wDQYDVQQDEwZWb29kb28wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQCdc7Qgx71o6Tq5dFUUhcCn8Nt-2Y9SGhm3WvsMYIqOIcHq3gjIKN9FWvXz
pBbTjz4lCwx-CJT82RBLNDFtsysfc0G7K_RsNKosYaM-L-DshO6R_314tptn9gnT
9tjTPXuiiICQlAP83BuTI148iEJWL36vbmv5AG6vrtk3T6ah5r2hBXQjt46sLQYw
eiM-peYIhPTIy9OYugogfqdzPvaJpDfAukqJBXqMxfscagKPYAGPaICKhobKr11a
Pam1Tchk2cBbtuYgSDz6ZGttsKE2omDbcmhbF7gBpRug-E2OH79Q4EVlSSoO9gZ6
AF4Km1A9uK9W_Pg8EPugY3Mgns6lAgMBAAGjJDAiMAsGA1UdDwQEAwIEMDATBgNV
HSUEDDAKBggrBgEFBQcDATANBgkqhkiG9w0BAQUFAAOCAQEACK9LQNkewOOugaYh
s4LfE3xdrRzrcaR0w5cf3wVcgR0ZZo98rDOtu3FAexpdh6vNaIdU4zAzNJPKKSso
3XF2LpQZovKIpUuN9pkZqslqZ0TLXqlyXMbheShcqIP1-m6qjZOp95N7jwgxBlEm
i_ne-rg1DicXFtAu90LpAZludaQGAyrj-LC37gzeMo2AG7BAuyFURXJFfxjpGmnu
euYfzZIMIQY-lNl6qm_vSMIz4uUKqq4lWndahnkJAwI2p5zUM0z3O6OMr_zr8eyr
dAL__H4NnG3gVyBbNoSbvbkxUt_C3oBwFFTupzRMQqJVjzbApyw5H0OzJPJKKkxx
hmIYTg"}]}}
```

The service validates the certificate according to the Omnibroker service policy.

```
[HTTP/1.1 OK Success
Content-Length: 81
Date: Mon, 19 May 2014 17:17:43 GMT
Server: Microsoft-HTTPAPI/2.0

{
  "ValidateResponse": {
    "Status": 200,
    "StatusDescription": "Success"}}
```

5.1.3. Credential Validation

The credential validation query provides certificate path validation and status checking.

The service provided by OBP is similar to that provided by OCSP and SCVP. Like SCVP, OBP is an agent selected by the relying party to validate certificates and/or construct trust paths on its behalf.

5.1.4. Message: QMessage

5.1.5. Message: QRequest

Every query request contains the following common elements:

Index :
: Integer [0..1] Index used to request a specific response when multiple responses are available.

5.1.6. Message: QResponse

Every Query Response contains the following common elements:

Status :
: Integer [1..1] Status return code value

StatusDescription :
: String [0..1] Describes the status code (ignored by processors)

Index :
: Integer [0..1] Index of the current response.

Count :
: Integer [0..1] Number of responses available.

5.1.7. Structure: Identifier

Specifies an Internet service by means of a DNS address and either a DNS service prefix, an IP port number or both. An Internet peer connection MAY be specified by additionally specifying an account.

Name :
: Name [1..1] The DNS name of the service to connect to. Internationalized DNS names MUST be encoded in punycode encoding.

Account :
: Label [0..1] Identifies the account to connect to in the case that a peer connection is to be established.

Service :
Name [0..1] The DNS service prefix defined for use with DNS records that take a service prefix including SRV.

Port :
Integer [0..1] IP Port number. A service identifier MUST specify either a service or a port or both.

5.1.8. Structure: Connection

IPAddress :
String [0..1] IP address in string representation

IPPort :
Integer [0..1] IP port. 1-65535

Transport :
String [0..1] Transport (RAW, TLS, IPSEC)

TransportPolicy :
String [0..1] Transport security policy as specified in [TBS]

ProtocolPolicy :
String [0..1] Application security policy specification as specified by the application protocol.

Advice :
Advice [0..1] Additional information that a service MAY return to support a service connection identification.

5.1.9. Structure: Credential

Type :
String [0..1] [TBS]

Data :
Binary [0..1] [TBS]

5.1.10. Structure: CertificateID

Type :
String [0..1] [TBS]

Data :
Binary [0..1] [TBS]

5.1.11. Structure: Advice

Additional information that a service MAY return to support a service connection identification. For example, DNSSEC signatures chains, SAML assertions, DANE records, Certificate Transparency proof chains,

etc.

Type :
Label [0..1] The IANA MIME type of the content type

Data :
Binary [0..1] The advice data.

5.1.12. Structure: Service

Describes a service connection

Identifier :
Identifier [0..Many] Internet addresses to which the service is to be bound.

Connection :
Connection [0..1] Service connection parameters.

6. OBPQuery

6.1. QueryConnect

Requests a connection context to connect to a specified Internet service or peer.

6.1.1. Message: QueryConnectRequest

Specifies the Internet service or peer that a connection is to be established to and the acceptable security policies.

Identifier :
Identifier [0..1] Identifies the service or peer to which a connection is requested.

Policy :
Label [0..Many] Acceptable credential validation policy.

ProveIt :
Boolean [0..1] If set the broker SHOULD send advice to permit the client to validate the proposed connection context.

6.1.2. Message: QueryConnectResponse

Returns one or more connection contexts in response to a QueryConnectRequest Message.

Connection :
Connection [0..Many] An ordered list of connection contexts with the preferred connection context listed first.

Advice :
Advice [0..1] Proof information to support the proposed connection context.

Policy :
Label [0..Many] Policy under which the credentials have been verified.

6.2. Validate

The Validate query requests validation of credentials presented to establish a connection. For example credentials presented by a server in the process of setting up a TLS session.

6.2.1. Message: ValidateRequest

Specifies the credentials to be validated and the purpose for which they are to be used.

Service :
Service [0..1] Describes the service for which the credentials are presented for access.

Credential :
Credential [0..Many] Credentials for which validation is requested.

CertificateID :
CertificateID [0..Many] OCSP Certificate Identifiers for which validation is requested.

Policy :
Label [0..Many] Policy under which the credentials have been verified.

6.2.2. Message: ValidateResponse

Reports the status of the credential presented.

Policy :
Label [0..Many] Policy under which the credentials have been verified.

7. Transport Bindings

To achieve an optimal balance of efficiency and availability, two transport bindings are defined:

JSON over HTTP (TLS or TCP)
Supports all forms of OBP transaction in all network environments.

JSON-B over UYFM (UDP)
Provides efficient support for all OBP query transactions and is accessible in most network environments.

Support for the HTTP binding is REQUIRED.

An OBP message consists of three parts:

Ticket [If required]
If specified, identifies the cryptographic key and algorithm parameters to be used to secure the message payload.

Payload [Required]
If the ticket context does not specify use of an encryption algorithm, contains the message data. Otherwise contains the message data encrypted under the encryption algorithm and key specified in the ticket context.

Authenticator [If required]
If the ticket context specifies use of a Message Authentication Code (MAC), contains the MAC value calculated over the payload data using the authentication key bound to the ticket.

Note that although each of the transport bindings defined in this specification entail the use of a JSON encoding for the message data, this is not a necessary requirement for a transport binding.

7.1. JSON Payload Binding

Integer
Data of type Integer is encoded using the JSON number encoding.

Name
Data of type Name is encoded using the JSON string encoding.

String
Data of type String is encoded using the JSON string encoding.

Binary
Data of type Binary is converted to strings using the Base64url encoding specified in [!RFC4648] /> and encoded using the JSON string type.

DateTime
Data of type DateTime is converted to string using the UTC time conversion specified in [!RFC3339] /> with a UTC offset of 00:00.

8. Acknowledgements

Rob Stradling, Robin Alden...

9. Security Considerations

9.1. Denial of Service

9.2. Breach of Trust

9.3. Coercion

10. IANA Considerations

[TBS list out all the code points that require an IANA registration]

11. Example Data

11.1. Ticket A

11.2. Ticket B

12. References

12.1. Normative References

[RFC6698] Hoffman, P.,Schlyter, J., "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.

[RFC5280] Cooper, D.,Santesson, S.,Farrell, S.,Boeyen, S.,Housley, R.,Polk, W., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5055] Freeman, T.,Housley, R.,Malpani, A.,Cooper, D.,Polk, W., "Server-Based Certificate Validation Protocol (SCVP)", RFC 5055, December 2007.

[RFC6962] Laurie, B.,Langley, A.,Kasper, E., "Certificate Transparency", RFC 6962, June 2013.

[I-D.hallambaker-omnipublish] , "[Reference Not Found!]".

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

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

[I-D.hallambaker-httpsession] Hallam-Baker, P, "HTTP Session Management", Internet-Draft draft-hallambaker-httpsession-02, 21 January 2014.

[I-D.hallambaker-wsconnect] Hallam-Baker, P, "JSON Service Connect (JCX) Protocol", Internet-Draft draft-hallambaker-wsconnect-05, 21 January 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[I-D.hallambaker-privatedns] Hallam-Baker, P, "Private-DNS", Internet-Draft draft-hallambaker-privatedns-00, 9 May 2014.

[RFC2616] Fielding, R.,Gettys, J.,Mogul, J.,Frystyk, H.,Masinter, L.,Leach, P.,Berners-Lee, T., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[I-D.hallambaker-jsonbcd] Hallam-Baker, P, "Binary Encodings for JavaScript Object Notation: JSON-B, JSON-C, JSON-D", Internet-Draft draft-hallambaker-jsonbcd-01, 21 January 2014.

Author’s Address

Phillip Hallam-Baker
Comodo Group Inc.

philliph@comodo.com