

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 3 April 2026

P. Hoffman
ICANN
30 September 2025

DNS Update with JSON
draft-hoffman-duj-04

Abstract

It is common for service providers such as certificate authorities and social media providers to want users to update the users' zones to prove that they control those zones, or to add other features. Currently, service providers tell users to do this using human language describing the resource record type and data values to enter into the zone. This document describes a text format, called "DNS update with JSON" or "DUJ", for such a service provider to give to a user, with the expectation that the user would copy and paste the text to their DNS operator to update the user's zone. DNS operators who know how to handle DUJ strings will make the update process easier and more predictable for their users.

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 3 April 2026.

Copyright Notice

Copyright (c) 2025 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
1.1. DUJ Use Case	3
1.2. DUJ Design	3
1.3. BCP 14 Language	4
2. Specification	5
2.1. DUJS	5
2.2. DUJ64	5
2.3. Notes on Owner Names	5
2.4. Notes on RRtypes	6
3. Processing	6
3.1. String Verification	6
3.2. Action Processing	7
4. IANA Considerations	7
5. Security Considerations	7
6. References	8
6.1. Normative References	8
6.2. Informative References	8
Appendix A. Acknowledgements	9
Author's Address	9

1. Introduction

There are many scenarios where someone gives instructions to a human to update their DNS zone in order to prove that they control those zones, or to add other features. For example, someone suggesting that you add an SPF [RFC7208] record to a zone might say "add a TXT record with the value "v=spf1 a:mail.yourname.example ip4:192.0.2.49" to your zone". The expectation is that you are either able to edit your zone file directly and understand the format of the records, or you use a DNS operator who has an interface for you to be able to add and modify records in your zone.

This document proposes a method for giving a human a string that can be copy-and-pasted to give a precise update for a zone. The protocol, called "DNS update with JSON" or "DUJ", specifies a string format in I-JSON [RFC7493] that represents the addition and deletion of records in a DNS zone. It is designed for DNS operators who accept strings in their interface for zone updates. The format self-identifies a string as DUJ, and the protocol lists what is and is not acceptable for DUJ strings. Using DUJ makes the update process more reliable for DNS operators and their users.

1.1. DUJ Use Case

DUJ is specifically targeted at improving the current common scenario of a user being told by an application service or a helpful friend to modify their DNS zone. DUJ is not intended for any automatic zone updates. DUJ is intended only for users who copy-and-paste.

There are two types of DUJ strings: DUJS and DUJ64. DUJS strings can be typed by users, but doing so can easily introduce errors that might negatively affect their DNS zone. DUJ64 strings cannot be safely typed by users.

There is no intention in the design for the user to change the text in a DUJ string. Because DUJ strings contain quoted text, and some typing systems might automatically unhelpfully convert quotation marks into "smart quotes", for some users typing DUJ strings might be impossible.

Different, more elaborate protocols for automatic updates, may be proposed separately. For example, DomainConnect ([DomainConnect], [I-D.kowalik-domainconnect]) defines an automated protocol that includes user affirmation before updates. DUJ is purposely more limited and less ambitious than those protocols, with the assumption that it will be much easier to deploy. Service providers might allow manual updates as they do today, manual updates by DUJ, and automated updates with a protocol like DomainConnect.

1.2. DUJ Design

This format is explicitly only meant for the use cases in Section 1.1. If the designer of an automated protocol is thinking of re-using DUJ in that protocol, they should not. DUJ is specifically designed for copy-and-paste by end users. It would be trivial to design a better format for describing automated DNS updates.

The design choice to use JSON arrays instead of objects is to increase security and reliability. This is to prevent key-value pairs to be added that might cause users or operators to possibly

process the DUJ strings incorrectly or to misinterpret them. For example, it is not possible to include comments in a DUJ string such as "For DKIM". The reason for this is that such comments could be used by an attacker to convince a user to make a change that they otherwise might not by adding a comment such as "Urgent security update".

DUJS strings are meant to be somewhat readable by the user. They might not understand what it says, but if they understand something about the DNS, they might. For example, you might see that a particular RRtype and Rdata are proposed to be added to your zone by looking at the DUJ string you are presented.

DUJ64 strings are purposely not readable by the user. However, the user can still see if records are being added or deleted.

DUJ strings should not be difficult for a service to create. There are cases described later where the quoting on the Rdata field can be tricky, but forcing an application to understand backslash quoting and apply it correctly is considered out of scope for a format this is only meant to replace human-readable instructions like "enter this record into your zone".

This document assumes that the application service will have looked in the user's zone before suggesting a zone update. That would likely be true in the current use case where the application service suggests an update to the user's zone.

Another explicit design for DUJ is that it is not extensible. If there is a reason to create a later version, the first string ("DJUS" or "DUJ64") can be changed to one that includes a new version identifier.

During the development of DUJ, there was a suggestion that each action string be followed by a list of record-data for that action. For example, to add two records, there would be only one action template of "add", which has two records. This suggested was rejected because it complicates the processing while saving only a tiny amount of space.

1.3. BCP 14 Language

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.

2. Specification

An example of a DUJS string is:

```
[ "DUJS", [ ["add", "mail.yourname.example TXT \v=spf1 a:mail.yourname.example ip4:192.0.2.49\" ] ] ]
```

An example of a DUJS string is:

```
[ "DUJ64", [ ["add", "bWFpbC55b3VybmFtZS5leGFtcGxlIFRYVCAidj1zcGYxIGE6bWFpbC55b3VybmFtZS5leGFtcGxlIGlwNDoxOTIuMC4yLjQ5Ig=="] ] ]
```

A DUJ string MUST be a JSON array with two values. The first value is the string "DUJS" or "DUJ64". The second value is an array, called the "update array", which lists all the updates. The update array MUST have a length of at least 1.

The arrays in the update array are called "action templates". Every action template is exactly two values: "action" and "record-data".

The action is a string specifying the action to be taken. The action string MUST be either "add" or "delete"; no other values are allowed.

The record-data is a string that is in the zone file format defined in [RFC1035]. The record-data MAY contain class names, as described in [RFC1035]. The record-data MAY contain time-to-live (TTL) values, as described in [RFC1035].

2.1. DUJS

The record-data in DUJS strings MUST NOT include zone file comments, directives, or embedded newlines from [RFC1035]. The record-data in DUJS strings MAY contain \DDD and \X escapes, and parentheses, from [RFC1035].

2.2. DUJ64

A DUJ64 string is identical to a DUJS string except that the record-data in a DUJ64 string is the Base64 [RFC4648] encoding of the record-data that would have gone into a DUJS string.

2.3. Notes on Owner Names

The owner name of a zone in a record-data string might be a name that does not yet exist because it is being created by an "add" action. A common example of this is adding an "underscore name" [RFC8552] such as "_smimecert" and "_xmpp". A DNS operator would have to determine whether the full owner name given in the record-data could be created, which might not be possible due to zone cuts.

The owner-name MUST NOT contain a wildcard.

2.4. Notes on RRtypes

The RRtype in the record-data with the name of the resource record type for the action. The RRtype's name is given in the "TYPE" column of the "Resource Record (RR) TYPES" IANA registry [Types-registry].

To specify an RRtype that is not yet in the registry, use the format specified in [RFC3597]. For example:

```
[ "DUJS", [ [ "add", "yourname.example TYPE4321 \# 4 0A000001" ] ] ]
```

3. Processing

This section defines rules for DNS operators who allow updates with DUJ to process DUJ strings they receive.

The update array is an ordered list of action templates. The DNS operator MUST process each action template in the order it appears in the update array. However, the DNS operator does so only after verifying that the entire DUJ string can be atomically applied to the target zone. The DNS operator MUST NOT process any action within the DUJ if any action would prevent the atomic application of the entire DUJ string.

The DNS operator MAY choose to skip an "add" action if it would create a record that is already in the zone. The DNS operator MAY choose to skip a "delete" action if it would delete a record that did not exist in the zone.

The DNS operator SHOULD be able to handle [RFC3597] RRtypes. However, they may have a local policy to not allow users to add or delete unknown RRtypes.

A DNS operator MAY reject any DUJ string for any reason, such as if it adds and then deletes the same record. If the DUJ was received from a user interface, the DNS operator SHOULD clearly describe why a DUJ was rejected.

3.1. String Verification

The DNS operator MUST not process any DUJ string where any of the following rules are not met:

- * The DUJ string MUST be valid I-JSON.
- * The first element of the DUJ string MUST be "DUJS" or "DUJ64".

- * The update array MUST have at least one action template.
- * Every action template MUST meet the following rules:
 - The action element MUST be either the exact string "add" or "delete"
 - The FQDN MUST be a valid fully-qualified domain name
 - The FQDN MUST NOT contain a wildcard
 - The RRtype MUST be recognized, or be in the format specified in [RFC3597]
 - The Rdata MUST be appropriate for the given RRtype

3.2. Action Processing

After verifying the DUJ string, the DNS operator processes each action template in order. When processing an action template, the DNS operator MUST verify:

- * that the user is authorized to change the zone named in the FQDN
- * that, for "delete" actions, that the exact record described in the action template exists
- * that, for "add" actions, that the exact record described in the action template does not already exist

A DNS operator SHOULD tell a user about every change made from a DUJ.

4. IANA Considerations

This document contains no actions for IANA.

5. Security Considerations

A DUJ has no cryptographic protection. It is, by design, only as secure as the current common scenario where a service tells a user to manually copy and paste some data (the RRtype and Rdata values) into an interface run by the DNS operator.

When a service gives the user a DUJ string, the authenticity of the source of the DUJ string and the integrity of the DUJ string is only as strong as the user's connection to the service. When a user pastes a DUJ string to a DNS operator, the authenticity of the source of the DUJ string and the integrity of the DUJS is only as strong as the user's connection to the DNS operator.

6. References

6.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <<https://www.rfc-editor.org/info/rfc3597>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/info/rfc7493>>.
- [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>>.

6.2. Informative References

- [RFC7208] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", RFC 7208, DOI 10.17487/RFC7208, April 2014, <<https://www.rfc-editor.org/info/rfc7208>>.
- [RFC8552] Crocker, D., "Scoped Interpretation of DNS Resource Records through "Underscored" Naming of Attribute Leaves", BCP 222, RFC 8552, DOI 10.17487/RFC8552, March 2019, <<https://www.rfc-editor.org/info/rfc8552>>.

[I-D.kowalik-domainconnect]

Kowalik, P., Blinn, A., Kolker, J., and S. Kerola, "Domain Connect Protocol - DNS provisioning between Services and DNS Providers", Work in Progress, Internet-Draft, draft-kowalik-domainconnect-02, 1 September 2025, <<https://datatracker.ietf.org/doc/html/draft-kowalik-domainconnect-02>>.

[DomainConnect]

"DomainConnect", <<https://www.domainconnect.org/>>.

[Types-registry]

"Domain Name System (DNS) Parameters", <<https://www.iana.org/assignments/dns-parameters/>>.

Appendix A. Acknowledgements

Andy Newton, Bob Harold, Gavin Brown, Jasdip Singh, John Levine, Libor Peltan, Pawel Kowalik, Peter Tomassen, and Robert Edmonds contributed substantial comments to early versions of this draft.

Author's Address

Paul Hoffman
ICANN
Email: paul.hoffman@icann.org