DNSOP Working Group
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

E. Bergstr旦m
L. Fernandez
J. Stenstam
The Swedish Internet Foundation
2 March 2026

Signalling Key State Via DNS EDNS(0) OPT
draft-berra-dnsop-keystate-02

Abstract

This document introduces the KeyState EDNS(0) option code, to enable the exchange of SIG(0) key state information between DNS entities via the DNS protocol. The KeyState option allows DNS clients and servers to include key state data in both queries and responses, facilitating mutual awareness of SIG(0) key status between child and parent zones. This mechanism addresses the challenges of maintaining synchronization of SIG(0) keys used for securing DNS UPDATE messages, thereby enhancing the efficiency and reliability of DNS operations that require coordinated key management.

This document proposes such a mechanism.

TO BE REMOVED: This document is being collaborated on in Github at: https://github.com/johanix/draft-berra-dnsop-opt-transaction-state (https://github.com/johanix/draft-berra-dnsop-transaction-state-00). The most recent working version of the document, open issues, etc, should all be available there. The authors (gratefully) accept pull requests.

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 September 2026.

Table of Contents

1. Introduction

In [I-D.ietf-dnsop-delegation-mgmt-via-ddns] a mechanism for automatic synchronization of delegation data between a child zone and a parent zone is proposed.

That mechanism relies on the parent validating and subsequently trusting the child SIG(0) public key so that the child is able to sign DNS UPDATE requests to the parent using the corresponding SIG(0) private key. While there is a mechanism for both uploading and rolling the public SIG(0) key there is still a risk of the child operator and the parent receiver getting out of sync.

This will be noticed by the child if a DNS UPDATE is refused. In this situation the parent UPDATE Receiver does have the opportunity and ability to provide more details of the refusal via an EDE opcode [RFC8914]. However, at that point it is too late to immediately get back in sync again and as a result the needed delegation synchronization that triggered the DNS UPDATE will be delayed until the child has managed to "re-bootstrap" a new SIG(0) key with the parent. As such re-bootstrapping may require manual actions, the length of the delay would not always be predictable.

The proposed new KeyState EDNS(0) Option addresses this problem by allowing the child and parent to exchange information about the current "state" of a SIG(0) key. This enables the child to become aware of any issue with the SIG(0) key currently being used in advance, and then address and resolve the problem. Additionally, the KeyState EDNS(0) Option enables the child to detect and resolve key synchronization issues before they cause operational failures.

1.1. Trusting SIG(0) Signed DNS Messages Between Child and Parent

Using the proposed OPT KeyState the child gains the ability to inform the parent about its own state and in return become aware of the parent’s state independently of a new DNS UPDATE (i.e. the OPT exchange may be sent via a normal DNS QUERY + response).

It should be pointed out that for the child to be able to trust the response from the parent, the response must be signed using a key that the child trusts. As the mechanism for automatic synchronization of delegation data aims to work independently of whether the involved zones are DNSSEC-signed or not, this requires that the parent UPDATE Receiver is able to sign the response using its own SIG(0) private key.

Hence there is a similar need for the UPDATE Receiver to "bootstrap" (as in "validate so that the key may be trusted") the child SIG(0) public key and for the child to "bootstrap" the UPDATE Receiver SIG(0) public key. The mechanism for doing this is described in [I-D.ietf-dnsop-delegation-mgmt-via-ddns].

Knowledge of DNS NOTIFY [RFC1996] and DNS Dynamic Updates [RFC2136] and [RFC3007] is assumed. DNS SIG(0) transaction signatures are documented in [RFC2931].

1.2. Requirements Notation

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. Terminology

SIG(0) An asymmetric signing algorithm that allows the recipient to only need to know the public key to verify a signature created by the sender’s private key.

3. Comparison to Extended DNS Errors [RFC8914]

EDE (Extended DNS Errors) specify a mechanism by which a receiver of a DNS message gains the ability to provide more information about the reason for a negative response. EDE data travels in an OPT record in the response and consist of an EDE code and, optionally, an EDE "Extra Text". It is possible to return EDE data with all types of DNS messages, including QUERY, NOTIFY and UPDATE.

However, there are three limitations to EDE that make it insufficient for communicating state between two parties:

1. An EDE must only be present in a response, not in the originating message.

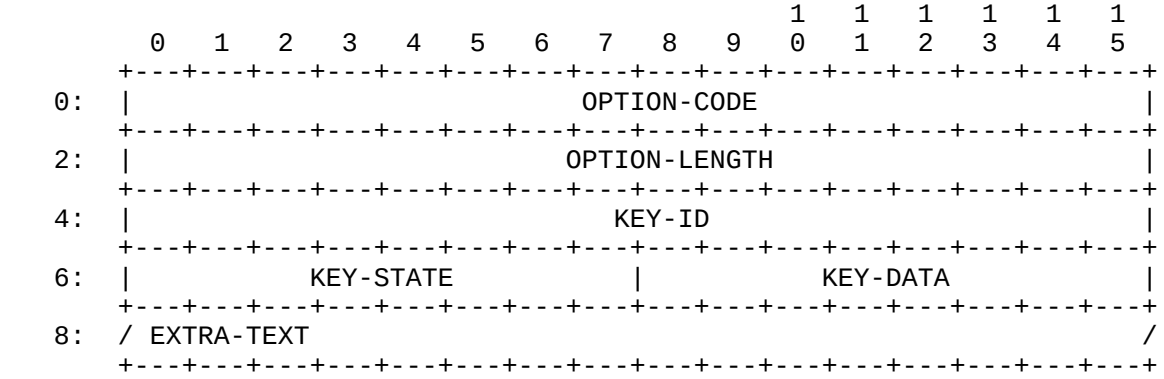
2. An EDE must only be used to augment an error response. It should not be part of a successful response.

3. An EDE must contain an EDE info code (16 bits) and may contain "Extra Text". However this extra text is intended for human consumption, not automated parsing. To communicate state between two parties this requirement is too strict.

These limitations are not a problem, as EDE serves a different purpose. But it is clear that for use cases like the above examples, EDE is not the right mechanism and another mechanism is needed. Hence the present proposal.

4. KeyState EDNS(0) Option Format

This document uses an Extended Mechanism for DNS (EDNS0) [RFC6891] option to include Key State information in DNS messages. The option is structured as follows:



Field definition details:

OPTION-CODE: 2 octets / 16 bits (defined in [RFC6891]) contains the value TBD for KeyState.

OPTION-LENGTH: 2 octets / 16 bits (defined in [RFC6891]) contains the length of the payload (everything after OPTION-LENGTH) in octets and should be 4 plus the length of the EXTRA-TEXT field (which may be zero octets long).

KEY-ID: 16 bits. The KeyID of the SIG(0) key that the KeyState inquiry is referring to. Note that while KeyIds are not guaranteed to be unique, it is the child that generates the initial SIG(0) key pair and all subsequent key pairs. Hence, it is the child’s responsibility to not use multiple keys with the same KeyId.

KEY-STATE: 8 bits. Currently defined values are listed in Section 6. Additional values may be defined in future documents.

KEY-DATA: 8 bits. Interpretation specific to each KEY-STATE.

EXTRA-TEXT: a variable-length sequence of octets that may hold additional information. This information is intended for human consumption (typically a reason or additional detail), not automated parsing. The length of the EXTRA-TEXT MUST be derived from the OPTION-LENGTH field. The EXTRA-TEXT field may be zero octets in length.

5. Use of the KeyState Option

The KeyState option may be included in outgoing message of type QUERY or UPDATE from the child to the UPDATE Receiver. The KeyState option MUST be present in such messages if the child supports the KeyState option.

The UPDATE Receiver MUST only include a KeyState option when responding to a DNS message that contained a KeyState option. I.e. the UPDATE Receiver must never assume that the child is able to interpret KeyState options. A KeyState option MAY be included in any type of response (SERVFAIL, NXDOMAIN, REFUSED, even NOERROR, etc.) to a query that includes a KeyState Option.

The KeyState option may always be ignored by the recipient. However, if the recipient does understand the KeyState option and responding with its own corresponding KeyState for the specified key make sense, then it is expected to do so.

This document includes a set of initial "key state" codepoints but is extensible via the IANA registry defined and created in Section 8.2.

6. Defined and Reserved Values for SIG(0) Key States

This document defines a number of initial KEY-STATE codes. The mechanism is intended to be extensible and additional KEY-STATE codes can be registered in the "KeyState Codes" registry (see Section 8.2). The KEY-STATE code from the "KeyState" EDNS(0) option is used to serve as an index into the "KeyState Option Codes" registry with the initial values defined below.

For KeyState signalling to be used the child includes a KeyState OPT in its query to indicate the ability to interpret a KeyState OPT in the response.

6.1. KeyStates Set By The Sender (the Child)

0: Automatic bootstrap requested. This assumes that the child SIG(0) public key is already published as a KEY record at the child apex.

1: Manual bootstrap requested. Child requests that manual bootstrap should be used (child does not want automatic bootstrap of the SIG(0) public key).

2: Key inquiry. Child requests information about current KeyState for specified key.

6.2. KeyStates Set By The UPDATE Receiver (the Parent or Its Agent)

4: SIG(0) key is known and trusted.

5: SIG(0) key is unknown.

6: SIG(0) key is invalid (eg. key data doesn’t match algorithm).

7: SIG(0) key is refused (eg. algorithm not accepted by policy).

8: SIG(0) key is known but validation has failed.

9: SIG(0) key is known but not trusted, automatic bootstrapping ongoing.

10: SIG(0) key is known but not trusted, manual bootstrapping required.

128-255: Reserved for private use.

To ensure that automatic delegation is correctly prepared and bootstrapped, the child (or an agent for the child) sends a DNS QUERY to the parent UPDATE Receiver with QNAME="child.parent." and QTYPE=ANY containing a KeyState OPT with KeyState-Code=2 and the KeyId of the SIG(0) key to inquire state for in the KEY-ID field.

The response should be signed by the SIG(0) key for the UPDATE Receiver and contain both the KeyId and the Key State encoded as described above.

7. Security Considerations

Key state signals in OPT queries and answers are unauthenticated unless the transaction carrying the state signal is secured via mechanisms such as [RFC2845], [RFC2931], [RFC8094] or [RFC8484]. Unauthenticated information MUST NOT be trusted as the state signals influence the DNS protocol processing. For instance, an attacker might cause a denial-of-service by forging a response claiming that the victim’s key is invalid, thereby halting the delegation synchronization procedure.

Moreover, it is assumed that the child has some means of validating messages from the parent during the initial phase when the child initializes the SIG(0) key synchronization. Otherwise, an attacker could prevent a child from initializing the synchronization by spoofing responses that refuse the key that the child is trying to upload. For that reason, it is expected that the parent has already published a public key that the child can use for this purpose. It could also possibly establish this trust out-of-band, such as via a physical meeting.

Lastly, SIG(0) transaction signatures are vulnerable to replay attacks, which could allow an attacker to disrupt the synchronization. Secure transport alternatives exist in [RFC8094] and [RFC8484].

8. IANA Considerations

8.1. New KeyState EDNS Option

This document defines a new EDNS(0) option, entitled "KeyState", assigned a value of TBD in the "DNS EDNS0 Option Codes (OPT)" registry

Value	Name	Status	Reference
TBD	KeyState	Standard	(This document)

Table 1

8.2. A New Registry for EDNS Option KeyState State Codes

The KeyState option also defines a 8-bit state field, for which IANA is requested to create and maintain a new registry entitled "KeyState Codes", used by the KeyState option. Initial values for the "KeyState Codes" registry are given below; future assignments in the 11-127 range are to be made through Specification Required review [BCP26].

KEY STATE	Mnemonic	Reference
0	REQUEST_AUTO_BOOTSTRAP	(This document)
1	REQUEST_MANUAL_BOOTSTRAP	(This document)
2	INQUIRY_KEY	(This document)
3	Unassigned	(This document)
4	KEY_TRUSTED	(This document)
5	KEY_UNKNOWN	(This document)
6	KEY_INVALID	(This document)
7	KEY_REFUSED	(This document)
8	KEY_VALIDATION_FAIL	(This document)
9	BOOTSTRAP_AUTO_ONGOING	(This document)
10	BOOTSTRAP_MANUAL_REQUIRED	(This document)
11-127	Unassigned	(This document)
128-255	Private use	(This document)

Table 2

9. References

9.1. Normative References

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <https://www.rfc-editor.org/rfc/rfc1996>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <https://www.rfc-editor.org/rfc/rfc2136>.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <https://www.rfc-editor.org/rfc/rfc2845>.

[RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <https://www.rfc-editor.org/rfc/rfc2931>.

[RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <https://www.rfc-editor.org/rfc/rfc3007>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <https://www.rfc-editor.org/rfc/rfc6891>.

[RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <https://www.rfc-editor.org/rfc/rfc8094>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/rfc/rfc8484>.

[RFC8914] Kumari, W., Hunt, E., Arends, R., Hardaker, W., and D. Lawrence, "Extended DNS Errors", RFC 8914, DOI 10.17487/RFC8914, October 2020, <https://www.rfc-editor.org/rfc/rfc8914>.

9.2. Informative References

[BCP26] Best Current Practice 26, <https://www.rfc-editor.org/info/bcp26>. At the time of writing, this BCP comprises the following:

Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[I-D.ietf-dnsop-delegation-mgmt-via-ddns]
Stenstam, J., Bergstr旦m, E., and L. Fernandez, "Automating DNS Delegation Management via DDNS", Work in Progress, Internet-Draft, draft-ietf-dnsop-delegation-mgmt-via-ddns-00, 17 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-delegation-mgmt-via-ddns-00>.

Appendix A. Change History (to be removed before publication)

* draft-berra-dnsop-opt-keystate-02

  Removed policy inquiry KeyState code 3 (INQUIRY_POLICY) and policy response codes 11 (POLICY_MANUAL_BOOTSTRAP_REQUIRED) and 12 (POLICY_AUTO_BOOTSTRAP). Bootstrap policy discovery is now handled entirely via the SVCB "bootstrap" SvcParamKey mechanism described in [I-D.ietf-dnsop-delegation-mgmt-via-ddns].

  Fixed wire format diagram: KEY-ID is 16 bits (a standard DNSSEC Key Tag), corrected byte offsets from 4/8/10 to 4/6/8.

  Replaced hardcoded section number references with kramdown anchors for stable cross-references.

* draft-berra-dnsop-opt-keystate-01

  Fixed minor typos.

* draft-berra-dnsop-opt-keystate-00

  Initial public draft.

Authors' Addresses

Erik Bergstr旦m
The Swedish Internet Foundation
Sweden
Email: erik.bergstrom@internetstiftelsen.se

Leon Fernandez
The Swedish Internet Foundation
Sweden

Email: leon.fernandez@internetstiftelsen.se

Johan Stenstam
The Swedish Internet Foundation
Sweden
Email: johan.stenstam@internetstiftelsen.se