Stream: Internet Engineering Task Force (IETF)
RFC: 9859
Category: Standards Track
Published: September 2025
ISSN: 2070-1721
Authors:
J. Stenstam
The Swedish Internet Foundation
P. Thomassen
deSEC, Secure Systems Engineering
J. Levine
Standcore LLC

RFC 9859
Generalized DNS Notifications

Abstract

This document generalizes and extends the use of DNS NOTIFY (RFC 1996) beyond conventional zone transfer hints to allow other types of actions that were previously lacking a trigger mechanism to be triggered via the DNS. Notifications merely nudge the receiver to initiate a predefined action promptly (instead of on a schedule); they do not alter the action itself (including any security checks it might employ).

To enable this functionality, a method for discovering the receiver endpoint for such notification messages is introduced, via the new DSYNC record type. Notification types are recorded in a new registry, with initial support for parental NS and DS record updates including DNSSEC bootstrapping.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9859.

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

The original DNS notifications [RFC1996], which are here referred to as "NOTIFY(SOA)", are sent from a primary server to a secondary server to minimize the latter's convergence time to a new version of the zone. This mechanism successfully addresses a significant inefficiency in the original protocol.

Today, similar inefficiencies occur in new use cases, in particular delegation maintenance (DS and NS record updates). Just as in the NOTIFY(SOA) case, a new set of notification types will have a major positive benefit by allowing the DNS infrastructure to completely sidestep these inefficiencies. For additional context, see Appendix A.

Although this document primarily deals with applying generalized notifications to the delegation maintenance use case, future extension for other applications (such as multi-signer key exchange) is possible.

No DNS protocol changes are introduced by this document. Instead, the mechanism makes use of a wider range of DNS messages allowed by the protocol.

Readers are expected to be familiar with DNSSEC [RFC9364], including [RFC6781], [RFC7344], [RFC7477], [RFC7583], [RFC8078], [RFC8901], and [RFC9615]. DNS-specific terminology can be found in [RFC9499].

1.1. Design Goals for Delegation Maintenance

When the parent operator is interested in notifications for delegation maintenance (such as DS or NS update hints), a service to accept these notifications will need to be made available. Depending on the context, this service may be run by the parent operator or by a designated entity who is in charge of handling the domain's delegation data (such as a domain registrar).

It seems desirable to minimize the number of steps that the notification sender needs to perform in order to figure out where to send the NOTIFY. This suggests that the lookup process be ignorant of the details of the parent-side relationships (e.g., whether or not there is a registrar).

This is addressed by parameterizing the lookup with the name of the child. The parent operator may then announce the notification endpoint in a delegation-specific way by publishing it at a child-specific name. (A catch-all endpoint may be indicated by wildcarding.)

The solution specified here is thus for the parent operator to publish the address where someone listens for notifications, in a child-specific way (see Section 3). Potential senders can easily determine the name of the parent and then look up that information (see Section 4.1).

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. DSYNC RR Type

This section defines the DSYNC RR type, which is subsequently used for discovering notification endpoints.

2.1. Wire Format

The DSYNC RDATA wire format is encoded as follows:

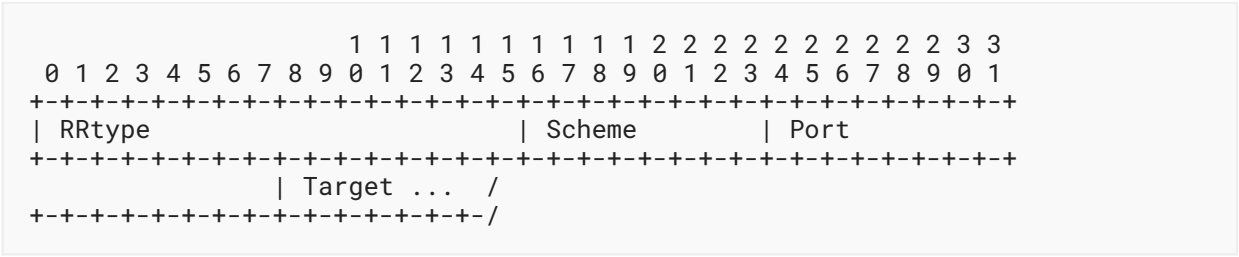

Figure 1: DSYNC RDATA Wire Format

RRtype: The type of generalized NOTIFY that this DSYNC RR defines the desired target address for (see the "Resource Record (RR) TYPEs" registry at <https://www.iana.org/assignments/dns-parameters/>). For now, only CDS and CSYNC are supported values, with the former indicating an updated CDS or CDNSKEY record set.

Scheme: The mode used for contacting the desired notification address. This is an 8-bit unsigned integer. Records with value 0 (null scheme) are ignored by consumers. Value 1 is described in this document, and values 128-255 are Reserved for Private Use. Other values are currently unassigned. Future assignments are maintained in the registry created in Section 6.2.

Port: The transport port number on the target host of the notification service. This is a 16-bit unsigned integer in network byte order. Records with value 0 are ignored by consumers.

Target: The fully-qualified, uncompressed domain name of the target host providing the service of listening for generalized notifications of the specified type. This name **MUST** resolve to one or more address records.

2.2. Presentation Format

The presentation format of the RDATA portion is as follows:

- The RRtype field is represented as a mnemonic from the "Resource Record (RR) TYPEs" registry.
- The Scheme field is represented by its mnemonic if assigned (see Section 6.2), and is otherwise represented as an unsigned decimal integer.
- The Port field is represented as an unsigned decimal integer.
- The Target field is represented as a <domain-name> (Section 5.1 of [RFC1035]).

2.3. Semantics

For now, the only scheme defined is 1 (mnemonic: NOTIFY). By publishing a DSYNC record with this scheme, a parent indicates that they would like child operators to send them a NOTIFY message (see Section 4) upon publication of a new CDS/CDNSKEY/CSYNC RRset to the address and port number that correspond to that DSYNC record, using conventional DNS transport [RFC1035].

Example (for the owner names of these records, see Section 3):

```
IN DSYNC  CDS   NOTIFY 5359 cds-scanner.example.net.
IN DSYNC  CSYNC NOTIFY 5360 csync-scanner.example.net.
```

Should a need for other mechanisms arise, other schemes may be defined to deal with such requirements using alternative logic.

Schemes are independent of the RRtype. They merely specify a method of contacting the target (whereas the RRtype is part of the notification payload).

3. Publication of Notification Targets

To use generalized notifications, it is necessary for the sender to know where to direct each NOTIFY message. This section describes the procedure for discovering that notification target.

Note that generalized NOTIFY messages are but one mechanism for improving the efficiency of automated delegation maintenance. Other alternatives, such as contacting the parent operator via an API or DNS Update [RFC2136], may (or may not) be more suitable in individual cases. Like generalized notifications, they similarly require a means for discovering where to send the API or DNS Update requests.

As the scope for the publication mechanism is wider than only to support generalized notifications, a unified approach that works independently of the notification method is specified in this section.

Parent operators participating in the discovery scheme for the purpose of delegation maintenance notifications **MUST** publish endpoint information using the record type defined in Section 2 under the `_dsync` subdomain of the parent zone, as described in the following subsections.

There **MUST NOT** be more than one DSYNC record for each combination of RRtype and Scheme. It is **RECOMMENDED** that zones containing DSYNC records be secured with DNSSEC.

For practical purposes, the parent operator **MAY** delegate the `_dsync` domain as a separate zone and/or synthesize records under it. If child-specificity is not needed, the parent can publish a static wildcard DSYNC record.

3.1. Wildcard Method

If the parent operator itself performs CDS/CDNSKEY or CSYNC processing for some or all delegations, or if the parent operator wants to relay notifications to some other party, a default notification target may be specified as follows:

```
*._dsync.example.  IN DSYNC  CDS   NOTIFY port target
*._dsync.example.  IN DSYNC  CSYNC NOTIFY port target
```

To accommodate indirect delegation management models, the designated notification target may relay notifications to a third party (such as the registrar, in ICANN's model). The details of such arrangements are out of scope for this document.

If for some reason the parent operator cannot publish wildcard records, the wildcard label may be dropped from the DSYNC owner name (i.e., it may be published at the `_dsync` label instead). This practice requires an additional step during discovery (see Section 4.1) and is therefore **NOT RECOMMENDED**.

3.2. Child-specific Method

It is also possible to publish child-specific records where the parent zone's labels are stripped from the child's Fully Qualified Domain Name (FQDN), and the result is used in place of the wildcard label.

As an example, consider a registrar offering domains like `child.example`, delegated from `example` zone. If the registrar provides the notification endpoint, e.g., `rr-endpoint.example:5300`, the parent may publish this information as follows:

```
child._dsync.example.  IN DSYNC  CDS NOTIFY 5300 rr-endpoint.example.
```

4. Delegation Maintenance: CDS/CDNSKEY and CSYNC Notifications

Delegation maintenance notifications address the inefficiencies related to scanning child zones for CDS/CDNSKEY records [RFC7344] [RFC8078] [RFC9615]. (For an overview of the issues, see Appendix A.)

NOTIFY messages for delegation maintenance **MUST** be formatted as described in [RFC1996], with the `qtype` field replaced as appropriate.

To address the CDS/CDNSKEY dichotomy, the NOTIFY(CDS) message (with `qtype=CDS`) is defined to indicate any child-side changes pertaining to an upcoming update of DS records. As the child DNS operator generally is unaware of whether the parent side consumes CDS records or prefers CDNSKEY, or when that policy changes, it seems advisable to publish both types of records, preferably using automation features of common authoritative nameserver software for ensuring consistency.

Upon receipt of NOTIFY(CDS), the parent-side recipient (typically, registry or registrar) **SHOULD** initiate the same DNS lookups and verifications for DNSSEC bootstrapping [RFC9615] or DS maintenance [RFC7344] [RFC8078] that would otherwise be triggered based on a timer.

The CSYNC [RFC7477] inefficiency may be similarly treated, with the child sending a NOTIFY(CSYNC) message (with `qtype=CSYNC`) to an address where the parent operator (or a designated party) is listening for CSYNC notifications.

In both cases, the notification will speed up processing times by providing the recipient with a hint that a particular child zone has published new CDS, CDNSKEY, and/or CSYNC records.

4.1. Endpoint Discovery

To locate the target for outgoing delegation maintenance notifications, the notification sender **MUST** perform the following steps:

1. Construct the lookup name by inserting the `_dsync` label after the first label of the delegation owner name.
2. Perform a lookup of type DSYNC for the lookup name, and validate the response if DNSSEC is enabled. If this results in a positive DSYNC answer, return it.

3. If the query resulted in a negative response:
   - If the response's SOA record indicates that the parent is more than one label away from the `_dsync` label, construct a new lookup name by inserting the `_dsync` label into the delegation owner name just before the parent zone labels inferred from the negative response. Then go to step 2.

     For example, assume that `subsub.sub.child.example` is delegated from `example` (and not from `sub.child.example` or `child.example`). The initial DSYNC query relating to it is thus directed at `subsub._dsync.sub.child.example`. This is expected to result in a negative response from `example`, and another query for `subsub.sub.child._dsync.example` is then required.
   - Otherwise, if the lookup name has any labels in front of the `_dsync` label, remove them to construct a new lookup name (such as `_dsync.example`). Then go to step 2. (This is to enable zone structures without wildcards.)
   - Otherwise, return null (no notification target available).

4.2. Sending Notifications

When creating or changing a CDS/CDNSKEY/CSYNC RRset in the child zone, the DNS operator **SHOULD** send a suitable notification to one of the endpoints discovered as described in Section 4.1.

A NOTIFY message can only carry information about changes concerning one child zone. When there are changes to several child zones, the sender **MUST** send a separate notification for each one.

When a primary name server publishes a new RRset in the child, there typically is a time delay until all publicly visible copies of the zone are updated. If the primary sends a notification at the exact time of publication, there is a potential for CDS/CDNSKEY/CSYNC processing to be attempted before the corresponding records are served. As a result, a desired update may not be detected (or appear inconsistent), preventing it from being applied.

Therefore, it is **RECOMMENDED** that the child would delay sending notifications to the recipient until a consistent public view of the pertinent records could be ensured.

4.2.1. Timeouts and Error Handling

NOTIFY messages are expected to elicit a response from the recipient ([RFC1996], Section 4.7). If no response is received, senders **SHOULD** employ the same logic as for SOA notifications ([RFC1996], Sections 3.5 and 3.6).

The recipient's attempt to act upon the delegation update request may fail for a variety of reasons (e.g., due to violation of the continuity requirement set forth in [RFC7344], Section 4.1). Such failures may occur asynchronously, even after the NOTIFY response has been sent.

In order to learn about such failures, senders **MAY** include an EDNS0 Report-Channel option [RFC9567] in the NOTIFY message to request that the receiving side report any errors by making a report query with an appropriate extended DNS error (EDE) code as described in [RFC8914]. (The prohibition of this option in queries ([RFC9567], Section 6.1) only applies to resolver queries and thus does not cover NOTIFY messages.)

When including this EDNS0 option, the second label (QTYPE) of the report query name is equal to the qtype received in the NOTIFY message. Its agent domain **MUST** be subordinate or equal to one of the NS hostnames, as listed in the child's delegation in the parent zone. This is to prevent malicious senders from causing the NOTIFY recipient to send unsolicited report queries to unrelated third parties.

For example, when receiving a NOTIFY(CDS) message for "example.com" with agent domain "errors.ns1.example.net", and when the requested DS update is found to break the delegation, then the following report query with EDE code 6 (DNSSEC Bogus) may be made (preferably over TCP):

```
qname: _er.59.example.com.6._er.errors.ns1.example.net.
qtype: TXT
```

4.2.2. Roles

While the CDS/CDNSKEY/CSYNC processing that follows the receipt of a NOTIFY will often be performed by the registry, the protocol anticipates that in some contexts (especially for ICANN gTLDs) registrars may take on the task. In such cases, the current registrar notification endpoint may be published, enabling notifications to be directed to the appropriate target. The mechanics of how this is arranged between registry and registrar are out of scope for this document; the protocol only offers the possibility to arrange things such that from the child perspective, how the parent-side parties are organized is inconsequential: Notifications are simply sent to the published address.

Because of the security model where a notification by itself never causes a change (it can only speed up the time until the next check for the same thing), the sender's identity is not crucial. This opens up the possibility of having an arbitrary party (e.g., a service separate from a nameserver) send the notifications, enabling this functionality even before the emergence of built-in support in nameserver software.

4.3. Processing of NOTIFY Messages for Delegation Maintenance

The following algorithm applies to NOTIFY(CDS) and NOTIFY(CSYNC) processing.

NOTIFY messages carrying notification payloads (records) for more than one child zone **MUST** be discarded, as sending them is an error.

Otherwise, upon receipt of a (potentially forwarded) NOTIFY message for a particular child zone at the published notification endpoint, the receiving side (parent registry or registrar) has two options:

1. Acknowledge receipt by sending a NOTIFY response as described in [RFC1996], Section 4.7, and schedule an immediate check of the CDS/CDNSKEY/CSYNC RRsets published by that particular child zone (as appropriate for the type of NOTIFY received).

   If the NOTIFY message contains an EDNS0 Report-Channel option [RFC9567] with an agent domain subordinate or equal to one of the NS hostnames listed in the delegation, the processing party **SHOULD** report any errors occurring during CDS/CDNSKEY/CSYNC processing by sending a report query with an appropriate EDE code as described in [RFC8914]. Reporting may be done asynchronously (outside of the NOTIFY transaction).

   When using periodic scanning, notifications preempt the scanning timer. If the NOTIFY-induced check finds that the CDS/CDNSKEY/CSYNC RRset is indeed new or has changed, the corresponding child's timer may be reset and the scanning frequency reduced (e.g., to once a week). If a CDS/CDNSKEY/CSYNC change is later detected through scanning (without having received a notification), the NOTIFY-related state **SHOULD** be cleared, reverting to the default scanning schedule for this child.

   When introducing CDS/CDNSKEY/CSYNC scanning support at the same time as NOTIFY support, backwards compatibility considerations regarding the scanning interval do not apply; a low-frequency scanning schedule **MAY** thus be used by default in such cases.

2. Do not act upon the notification. To prevent retries, recipients **SHOULD** acknowledge the notification by sending a NOTIFY response even when otherwise ignoring the request, combined with a report query if feasible (see above). One reason to do this may be a rate limit (see Section 5), in which case "Blocked" (15) may be a suitable extended DNS error code.

Implementing the first option will significantly decrease the convergence time (between publication of a new CDS/CDNSKEY/CSYNC record in the child and publication of the resulting DS), thereby providing improved service for the child.

If, in addition to scheduling an immediate check for the child zone of the notification, the scanning schedule is also modified to be less frequent, the cost of providing the scanning service will be reduced.

5. Security Considerations

If an action is triggered by the receipt of a DNS NOTIFY, its execution relies on the same security model that the receiving party would apply if the action were triggered by something else. This is because the notification affects the action's timing alone. For example, DS bootstrapping is expected to be performed the same way, independently of the type of trigger; this includes all security and authentication requirements (e.g., [RFC9615]) that the parent registry/registrar has chosen to apply.

The original NOTIFY specification sidesteps most security issues by not relying on the information in the NOTIFY message in any way and instead only using it to "enter the state it would if the zone's refresh timer had expired" (Section 4.7 of [RFC1996]).

This security model is reused for generalized NOTIFY messages. Therefore, it seems impossible to affect the behaviour of the recipient of the NOTIFY other than by hastening the timing for when different checks are initiated. As a consequence, while notifications themselves can be secured via access control mechanisms, this is not a requirement.

In general, the receipt of a notification message will cause the receiving party to perform one or more outbound queries for the records of interest (for example, NOTIFY(CDS) will cause CDS/CDNSKEY queries). When done using standard DNS, the size of these queries is comparable to that of the NOTIFY messages themselves, rendering any amplification attempts futile. The number of queries triggered per notification is also limited by the requirement that a NOTIFY message can refer to one child only.

However, when the outgoing query occurs via encrypted transport, some amplification is possible, both with respect to bandwidth and computational burden. In this case, the usual principle of bounding the work applies, even under unforeseen events.

Therefore, receivers **MUST** implement rate limiting for notification processing. It is **RECOMMENDED** to configure rate limiting independently for both the notification's source IP address and the name of the zone that is conveyed in the NOTIFY message. Rate limiting also mitigates the processing load from garbage notifications.

Alternative solutions (such as signing notifications and validating their signatures) appear significantly more expensive without tangible benefit.

In order to facilitate schemes that are authenticated outside of DNSSEC (such as via SIG(0)), zones containing DSYNC records are not required to be signed. Spoofed DSYNC responses would prevent notifications from reaching their legitimate target, and a different party may receive unsolicited notifications; however, both effects can also be achieved in the presence of DNSSEC. The illegitimate target is also enabled to learn notification contents in real time, which may be a privacy concern for the sender. If so, the sender may choose to ignore unsigned DSYNC records.

6. IANA Considerations

6.1. DSYNC RR Type

IANA has registered DSYNC in the "Resource Record (RR) TYPEs" registry under the "Domain Name System (DNS) Parameters" registry group as follows:

Type: DSYNC
Value: 66
Meaning: Endpoint discovery for delegation synchronization
Reference: RFC 9859

6.2. DSYNC Scheme Registration

IANA has created the following new registry in the "Domain Name System (DNS) Parameters" registry group:

Name: DSYNC: Location of Synchronization Endpoints
Registration Procedure: Expert Review
Reference: RFC 9859

The initial contents for the registry are as follows:

RRtype	Scheme (Mnemonic)	Purpose	Reference
	0	Null scheme (no-op)	RFC 9859
CDS	1 (NOTIFY)	Delegation management	RFC 9859
CSYNC	1 (NOTIFY)	Delegation management	RFC 9859
	2-127	Unassigned	
	128-255	Reserved for Private Use	RFC 9859

Table 1

Requests to register additional entries **MUST** include the following fields:

RRtype: An RRtype that is defined for use
Scheme: The mode used for contacting the desired notification address
Mnemonic: The scheme's shorthand string used in presentation format
Purpose: Use case description
Reference: Location of specification or registration source

Registration requests are to be recorded by IANA after Expert Review [RFC8126]. Expert Reviewers should take the points below into consideration; however, they are experts and should be given substantial latitude:

- Point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments. The code points tagged as "Private Use" are intended for testing purposes and closed environments. Code points in other ranges should not be assigned for testing.
- A specification of a scheme is desirable, but early assignment before a specification is available is also possible. When specifications are not provided, the description provided needs to have sufficient information to identify what the point is being used for. A scheme number may have exactly one mnemonic.

- Experts should take into account that field values are fit for purpose. For example, the mnemonic should be indicative and have a plausible connection to the scheme's notification mechanism.

6.3. _dsync Underscore Name

Per [RFC8552], IANA has registered the following entries to the "Underscored and Globally Scoped DNS Node Names" registry within the "Domain Name System (DNS) Parameters" registry group:

RR Type	_NODE NAME	Reference
DSYNC	_dsync	RFC 9859

Table 2

7. References

7.1. 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>.

[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/info/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/info/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/info/rfc2136>.

[RFC7344] Kumari, W., Gudmundsson, O., and G. Barwood, "Automating DNSSEC Delegation Trust Maintenance", RFC 7344, DOI 10.17487/RFC7344, September 2014, <https://www.rfc-editor.org/info/rfc7344>.

[RFC7477] Hardaker, W., "Child-to-Parent Synchronization in DNS", RFC 7477, DOI 10.17487/RFC7477, March 2015, <https://www.rfc-editor.org/info/rfc7477>.

[RFC8078] Gudmundsson, O. and P. Wouters, "Managing DS Records from the Parent via CDS/CDNSKEY", RFC 8078, DOI 10.17487/RFC8078, March 2017, <https://www.rfc-editor.org/info/rfc8078>.

[RFC8126] 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>.

[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>.

[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>.

[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/info/rfc8914>.

[RFC9364] Hoffman, P., "DNS Security Extensions (DNSSEC)", BCP 237, RFC 9364, DOI 10.17487/RFC9364, February 2023, <https://www.rfc-editor.org/info/rfc9364>.

[RFC9499] Hoffman, P. and K. Fujiwara, "DNS Terminology", BCP 219, RFC 9499, DOI 10.17487/RFC9499, March 2024, <https://www.rfc-editor.org/info/rfc9499>.

[RFC9567] Arends, R. and M. Larson, "DNS Error Reporting", RFC 9567, DOI 10.17487/RFC9567, April 2024, <https://www.rfc-editor.org/info/rfc9567>.

[RFC9615] Thomassen, P. and N. Wisiol, "Automatic DNSSEC Bootstrapping Using Authenticated Signals from the Zone's Operator", RFC 9615, DOI 10.17487/RFC9615, July 2024, <https://www.rfc-editor.org/info/rfc9615>.

7.2. Informative References

[RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <https://www.rfc-editor.org/info/rfc6781>.

[RFC7583] Morris, S., Ihren, J., Dickinson, J., and W. Mekking, "DNSSEC Key Rollover Timing Considerations", RFC 7583, DOI 10.17487/RFC7583, October 2015, <https://www.rfc-editor.org/info/rfc7583>.

[RFC8901] Huque, S., Aras, P., Dickinson, J., Vcelak, J., and D. Blacka, "Multi-Signer DNSSEC Models", RFC 8901, DOI 10.17487/RFC8901, September 2020, <https://www.rfc-editor.org/info/rfc8901>.

Appendix A. Efficiency and Convergence Issues in DNS Scanning

A.1. Original NOTIFY for Zone Transfer Nudging

[RFC1996] introduced the concept of a DNS NOTIFY message, which was used to improve the convergence time for secondary servers when a DNS zone had been updated in the primary server. The basic idea was to augment the original "pull" mechanism (a periodic SOA query) with a "push" mechanism (a NOTIFY) for a common case that was otherwise very inefficient (due to either slow convergence or wasteful and overly frequent scanning of the primary for changes).

While it is possible to indicate how frequently checks should occur (via the SOA Refresh parameter), these checks did not allow catching zone changes that fall between checkpoints. [RFC1996] addressed the optimization of the time-and-cost trade-off between a secondary server frequently checking for new versions of a zone and infrequent checks by replacing scheduled scanning with the more efficient NOTIFY mechanism.

A.2. Similar Issues for DS Maintenance and Beyond

Today, we have similar issues with slow updates of DNS data in spite of the data having been published. The two most obvious cases are CDS and CSYNC scanners deployed in a growing number of TLD registries. Because of the large number of child delegations, scanning for CDS and CSYNC records is rather slow (as in, infrequent).

Only a very small number of the delegations will have updated a CDS or CDNSKEY record in between two scanning runs. However, frequent scanning for CDS and CDNSKEY records is costly, and infrequent scanning causes slower convergence (i.e., delay until the DS RRset is updated).

Unlike in the original case, where the primary is able to suggest the scanning interval via the SOA Refresh parameter, an equivalent mechanism does not exist for DS-related scanning.

All of the above also applies to automated NS and glue record maintenance via CSYNC scanning [RFC7477]. Again, given that CSYNC records change only rarely, frequent scanning of a large number of delegations seems disproportionately costly, while infrequent scanning causes slower convergence (delay until the delegation is updated).

While use of the NOTIFY mechanism for coordinating the key exchange in multi-signer setups [RFC8901] is conceivable, the detailed specification is left for future work.

Acknowledgements

The authors acknowledge the contributions and reviews of the following individuals (in order of their first contribution or review): Joe Abley, Mark Andrews, Christian Elmerot, Ólafur Guðmundsson, Paul Wouters, Brian Dickson, Warren Kumari, Geoff Huston, Patrick Mevzek, Tim

Wicinski, Q Misell, Stefan Ubbink, Matthijs Mekking, Kevin P. Fleming, Nicolai Leymann, Giuseppe Fioccola, Peter Yee, Tony Li, Paul Wouters, Roman Danyliw, Peter van Dijk, John Scudder, Éric Vyncke, and Oli Schacher.

Authors' Addresses

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

Peter Thomassen
deSEC, Secure Systems Engineering
Email: peter@desec.io

John Levine
Standcore LLC
Email: standards@standcore.com