# Automating DNS Parent—Child Synchronization

Johan Stenstam

September 4, 2023

# Automating DNS Parent—Child Synchronization

. . . without scanners?

Johan Stenstam

September 4, 2023

## Problem Statement

Since the inception of time

- i.e. when DNS was invented and life as we know it begun

there has been a problem with the delegation information (the NS RRset and sometimes the glue records) getting out of sync between the (authoritative) data in the child zone and the delegation in the parent zone.

- Slowly, over time, updates happen that are not synched with the parent. For all sorts of reasons.
- We're beginning to make some inroads on automatic synchronization for a subset of parent/child cases via so-called CDS and CSYNC scanners run by the parent.
- But most of the DNS name space is outside of that subset

INTERNET STIFTELSEN

## What Are The Use Cases?

While the risk of “parent and child getting out of sync” exists for all zone cuts, the risk is clearly not the same everywhere.

- Some registries do periodic, proactive scans of all delegations, trying to catch errors before they cause problems.
- “Corporate environments” in many cases have all zone management under control of their IPAM-systems, that can keep things in sync.

So perhaps there is no problem to be solved?

## Use Cases, cont'd

But even so, there are lots of cases when zone cuts cross organisational borders and are therefore more difficult to maintain.

- Academia and the educational sector in general.
- The health care sector: lots of hopitals with lots of departments.
- Government agencies and departments.
- A gazillion zones that are important to the owner, but DNS is not a focus, DNSSEC isn't used and the parent is not proactive.

INTERNET STIFTELSEN

## Use Cases, cont'd

But even so, there are lots of cases when zone cuts cross organisational borders and are therefore more difficult to maintain.

- Academia and the educational sector in general.
- The health care sector: lots of hopitals with lots of departments.
- Government agencies and departments.
- A gazillion zones that are important to the owner, but DNS is not a focus, DNSSEC isn't used and the parent is not proactive.

In short, many, many places where DNS is not a core focus, nor a core expertise.

- It would be great if we could find a way to keep such delegations in sync automatically.

INTERNET STIFTELSEN

# Let's Talk About Scanners

I argue that CDS and CSYNC scanners are a kludge that to some extent mostly represent a failure to come up with a clean solution.

- **Scanners are inefficient**. Oceans of effort for very little (albeit important) gain.
- **Scanners are slow**. It is bad service to the customers (the children) that it usually takes 24h or so between scanner runs.
- **Scanners require** the children to have **DNSSEC** deployed.
- **Scanners are complex**. It is yet another service ro run and maintain. They require complicated logic.
    - It seems unlikely that scanners will ever be a popular choice outside of the registry space.
    - I.e. they represent a partial solution that only work for some, not for all zones. That's always bad.

But, given their drawbacks, they do work, and that's why a growing number of TLDs (and some registrars) are deploying scanners.

INTERNET
STIFTELSEN

## Let's Improve The Scanner Efficiency

Peter Thomassen, John Levine and I have worked quite a lot on so-called "generalised DNS notifications" (see `draft-thomassen-generalised-notify-NN`).

- The general idea is trivial: when the child publishes a `CDS` record it also sends a DNS NOTIFY(CDS) to the parent, as a hint that it would be a good idea to scan this particular child for `CDS`, as it has just changed.
- Same thing for `CSYNC`: on publication of a new `CSYNC`, also send a NOTIFY(CSYNC) to the parent.

Modulo some details, like where to send the notifications, this works fine and does indeed much improve speed of convergence (i.e. a published new `CDS` or `CSYNC` may cause a parent scan within seconds).

- It does not, however, address the complexity issues with running scanners.

## Where To Send The Generalised Notifications

How does the child know where the notifications should be sent?

There are several alternatives, including:

- Static configuration (typically in the child primary nameserver)
- Dynamic lookup of the location of the parent notification receiver:

```
parent.  IN SOA ...
...
parent.  IN NOTIFY CDS    1  5301  notifications.parent.
parent.  IN NOTIFY CSYNC  1  5302  notifications.parent.
```

- Notifications are typically sent to a "service", not to the parent primary nameserver directly. The notification is merely a hint to suggest "please scan this child ASAP".
- The "scheme" parameter is there to allow different mechanisms. "1" is interpreted as "send a a NOTIFY for the right RRtype to the target and port specified.

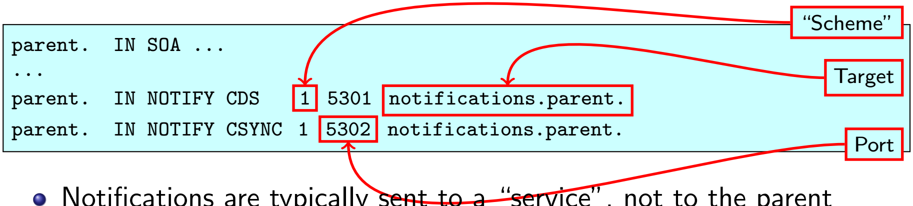
INTERNET STIFTELSEN

## Where To Send The Generalised Notifications

How does the child know where the notifications should be sent?

There are several alternatives, including:

- Static configuration (typically in the child primary nameserver)
- Dynamic lookup of the location of the parent notification receiver:



```
parent.  IN SOA ...
...
parent.  IN NOTIFY CDS    1 5301 notifications.parent.
parent.  IN NOTIFY CSYNC  1 5302 notifications.parent.
```

- Notifications are typically sent to a "service", not to the parent primary nameserver directly. The notification is merely a hint to suggest "please scan this child ASAP".
- The "scheme" parameter is there to allow different mechanisms. "1" is interpreted as "send a a NOTIFY for the right RRtype to the target and port specified.

## There Are Different Parent/Child Combinations

Part of the problem is that there are a number of different possible combinations of parent and child abilities.

- We want to find solutions that enable anyone who so choses to have automatic delegation synchronisation.
- One parameter is clearly whether DNSSEC is in use or not (CDS and CSYNC scanners depend on DNSSEC for validation).
- Another is whether the parent is a registry or not. The term "registry" is used to cover the spectrum of parents that are more complex in various ways.

| Types of parents | Signed children | Unsigned children |
|---|---|---|
| "Registries" | ? | ? |
| Other parents | ? | ? |

INTERNET STIFTELSEN

## There Are Different Parent/Child Combinations

Part of the problem is that there are a number of different possible combinations of parent and child abilities.

- We want to find solutions that enable anyone who so choses to have automatic delegation synchronisation.
- One parameter is clearly whether DNSSEC is in use or not (CDS and CSYNC scanners depend on DNSSEC for validation).
- Another is whether the parent is a registry or not. The term "registry" is used to cover the spectrum of parents that are more complex in various ways.

They may be regulated by external contracts

There may be registrars that are responsible for the child data

| Types of parents | Signed children | Unsigned children |
|---|---|---|
| "Registries" | ? | ? |
| Other parents | ? | ? |

INTERNET STIFTELSEN

## There Are Different Parent/Child Combinations

Part of the problem is that there are a number of different possible combinations of parent and child abilities.

- We want to find solutions that enable anyone who so choses to have automatic delegation synchronisation.
- One parameter is clearly whether DNSSEC is in use or not (CDS and CSYNC scanners depend on DNSSEC for validation).
- Another is whether the parent is a registry or not. The term "registry" is used to cover the spectrum of parents that are more complex in various ways.

They may be regulated by external contracts

There may be registrars that are responsible for the child data

On the other hand, they are usually better resourced

| Types of parents | Signed children | Unsigned children |
|---|---|---|
| "Registries" | ? | ? |
| Other parents | ? | ? |

INTERNET STIFTELSEN

## There Are Different Parent/Child Combinations

Part of the problem is that there are a number of different possible combinations of parent and child abilities.

- We want to find solutions that enable anyone who so choses to have automatic delegation synchronisation.
- One parameter is clearly whether DNSSEC is in use or not (CDS and CSYNC scanners depend on DNSSEC for validation).
- Another is whether the parent is a registry or not. The term "registry" is used to cover the spectrum of parents that are more complex in various ways.

| Types of parents | Signed children | Unsigned children |
|---|---|---|
| "Registries" | OK | ? |
| Other parents | ? | ? |

Scanners + generalized notifications work here

INTERNET STIFTELSEN

## Synchronisation Without A Scanner?

Could it be possible to find a design where we get the automated synchronisation between child and parent without the parent having to operate a scanner?

**Yes.**

Such a design already exists, but has not seen much use yet. It's based on DDNS (i.e. DNS Dynamic Updates).

- However, it is not based on DDNS using TSIG keys (that have to be stored in the configuration and therefore doesn't scale to large numbers of children).
- Instead it is based on using so-called `SIG(0)` signatures for the DDNS updates for security.
- An additional feature is that the SIG(0) validation mechanism works **independently of DNSSEC**.

INTERNET
STIFTELSEN

## Dynamic Update Policies For Many Delegations

Assume delegations that use in-bailiwick nameservers, a la

```
foo.parent.       IN NS ns1.foo.parent.
foo.parent.       IN NS ns2.foo.parent.
ns1.foo.parent.   IN A 1.2.3.4
ns1.foo.parent.   IN AAAA 2001:a:bad:cafe::53
ns2.foo.parent.   IN A 2.3.4.5
```

In one implementation (BIND9) it is then possible to use an update policy a la:

```
update-policy {
    grant *  selfsub  *  NS A AAAA  KEY ;
};
```

# Dynamic Update Policies For Many Delegations

Assume delegations that use in-bailiwick nameservers, a la

```
foo.parent.        IN NS ns1.foo.parent.
foo.parent.        IN NS ns2.foo.parent.
ns1.foo.parent.    IN A 1.2.3.4
ns1.foo.parent.    IN AAAA 2001:a:bad:cafe::53
ns2.foo.parent.    IN A 2.3.4.5
```

In one implementation (BIND9) it is then possible to use an update policy a la:

```
update-policy {
    grant * selfsub * NS A AAAA KEY ;
};
```

For a match, both wildcards must expand to the same string

Only these RR types

Exact match and subdomains of match

# Dynamic Update Policies For Many Delegations

Assume delegations that use in-bailiwick nameservers, a la

```
foo.parent.       IN NS ns1.foo.parent.
foo.parent.       IN NS ns2.foo.parent.
ns1.foo.parent.   IN A 1.2.3.4
ns1.foo.parent.   IN AAAA 2001:a:bad:cafe::53
ns2.foo.parent.   IN A 2.3.4.5
```

For a match, both wildcards must expand to the same string

Only these RR types

In one implementation (BIND9) it is then possible to use an update policy a la:

```
update-policy {
    grant * selfsub * NS A AAAA  KEY;
};
```

Exact match and subdomains of match

This policy would allow a key with the name `foo.parent.` to update records like `foo.parent. NS ...` and `ns1.foo.parent. A ...` but not any records that do not have an owner name ending in `foo.parent.`.

- I.e. it is impossible to affect any delegation except "your own".

## Dynamic Update Policies For Many Delegations

Assume delegations that use in-bailiwick nameservers, a la

```
foo.parent.       IN NS ns1.foo.parent.
foo.parent.       IN NS ns2.foo.parent.
ns1.foo.parent.   IN A 1.2.3.4
ns1.foo.parent.   IN AAAA 2001:a:bad:cafe::53
ns2.foo.parent.   IN A 2.3.4.5
```

For a match, both wildcards must expand to the same string

Only these RR types

In one implementation (BIND9) it is then possible to use an update policy a la:

```
update-policy {
    grant * selfsub * NS A AAAA KEY;
};
```

Allow the child to roll the key

Exact match and subdomains of match

This policy would allow a key with the name `foo.parent.` to update records like `foo.parent. NS ...` and `ns1.foo.parent. A ...` but not any records that do not have an owner name ending in `foo.parent.`.

- I.e. it is impossible to affect any delegation except "your own".

## DDNS Updates? Are you insane?

That's an ongoing discussion, and, AFAIK, the jury is still out.

But bear with me for a bit.

It is important to figure out **why DDNS has not been used much for delegation synchronisation**.

- After all, the functionality has been there for years.

I argue that the reason is a combination of two factors.

## Problems Using DDNS For Synchronisation

- **Problem #1:** It is difficult to figure out where to send the dynamic update. DDNS is mostly used inside an organisation, but in the parent/child case there is often a need to cross an organisational boundary.
  - The parent primaries are often hidden, but even if it is known, it may be difficult to reach because it is locked down behind firewalls, etc. This would require "update forwarding", which is very icky.
- **Problem #2:** The assumption has been that the DDNS update is sent to a **nameserver**, and, if accepted, is immediately applied to the zone.
  - This is not acceptable to many parent zones. They have requirements on applying policies and safety checks (not to mention audit trails) on any data before it is added to the zone.

But. . . both of these issues are addressed in the Internet-Draft about generalised notifications. **Let's combine the two!**

INTERNET
STIFTELSEN

## Combining Generalised Notifications and DDNS Updates

The notification draft uses a parameter called "scheme" to indicate possible different methods to "notify the parent":

"NOTIFY scheme"

Where to send notifications

```
parent.  IN SOA ...
...
parent.  IN NOTIFY CDS    1  5301 notifications.parent.
parent.  IN NOTIFY CSYNC 1  5302 notifications.parent.
```

Let's add a new scheme for the DDNS-based synchronisation:

INTERNET STIFTELSEN

## Combining Generalised Notifications and DDNS Updates

The notification draft uses a parameter called “scheme” to indicate possible different methods to “notify the parent”:

“NOTIFY scheme”

Where to send notifications

```
parent.   IN SOA ...
...
parent.   IN NOTIFY CDS    1  5301 notifications.parent.
parent.   IN NOTIFY CSYNC  1  5302 notifications.parent.
```

Let’s add a new scheme for the DDNS-based synchronisation:

“DDNS scheme”

Where to send DDNS updates

```
parent.   IN NOTIFY ANY    2  5310  ddns-updates.parent.
```

Port for DDNS updates

INTERNET STIFTELSEN

## Combining Generalised Notifications and DDNS Updates

The notification draft uses a parameter called "scheme" to indicate possible different methods to "notify the parent":

"NOTIFY scheme"

Where to send notifications

```
parent.   IN SOA ...
...
parent.   IN NOTIFY CDS    1  5301  notifications.parent.
parent.   IN NOTIFY CSYNC  1  5302  notifications.parent.
```

Let's add a new scheme for the DDNS-based synchronisation:

"DDNS scheme"

Where to send DDNS updates

```
parent.   IN NOTIFY ANY    2  5310  ddns-updates.parent.
```

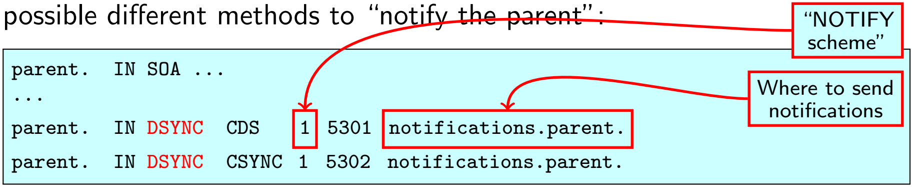
Port for DDNS updates

That's works, but. . . there is a naming confusion here.

- The mnemonic NOTIFY doesn't work well for DDNS updates.
- But what should be used instead? Not my decision, but let me just throw out a suggestion, based on the topic being "**d**elegation **sync**hronisation".

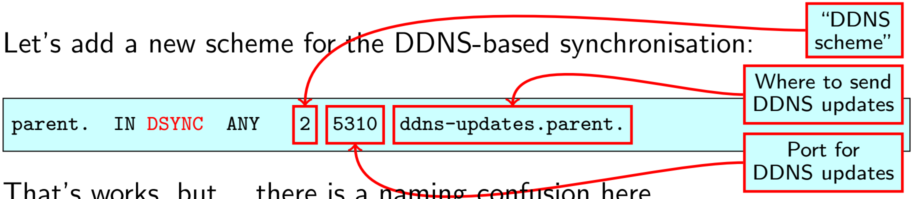
## Combining Generalised Notifications and DDNS Updates

The notification draft uses a parameter called "scheme" to indicate possible different methods to "notify the parent":



```
parent.   IN SOA ...
...
parent.   IN DSYNC   CDS    1  5301 notifications.parent.
parent.   IN DSYNC   CSYNC  1  5302 notifications.parent.
```

Let's add a new scheme for the DDNS-based synchronisation:



```
parent.   IN DSYNC   ANY    2  5310  ddns-updates.parent.
```

That's works, but. . . there is a naming confusion here.

- The mnemonic DSYNC would work well for DDNS updates.
- But what should be used instead? Not my decision, but let me just throw out a suggestion, based on the topic being "**d**elegation **sync**hronisation".

## Let's Summarise

- The "NOTIFY scheme" and the "DDNS scheme" are **alternate** methods for parent synchronisation.
  - I see no reason to expect the same parent to support more than one method (or zero methods).
- The same mechanism is used for signalling support for generalised notifications (and where to send them) and support for DDNS updates (and where to send them)
  - This addresses the first major issue with using DDNS for synchronisation.
- By sending the DDNS update to a "service" rather than to the parent primary nameserver, the parent may implement whatever policies and checks that it wants before applying the update.
  - This addresses the second major issue with using DDNS updates

INTERNET STIFTELSEN

## A New Look At the Different Cases

Scanner + generalized notifications work

| Types of parents | Signed children | Unsigned children |
|---|---|---|
| "Registries" | OK | ? |
| Other parents | ? | ? |

An important detail is that the SIG(0) keys **do not have to be present in the parent zone**. That's an artifact of the BIND9 implementation.

- The SIG(0) public keys only have to be available to the DDNS receiver.

INTERNET STIFTELSEN

## A New Look At the Different Cases

Scanner + generalized notifications work

| Types of parents | Signed children | Unsigned children |
|---|---|---|
| "Registries" | OK | ? |
| Other parents | OK | ? |

An important detail is that the SIG(0) keys **do not have to be present in the parent zone**. That's an artifact of the BIND9 implementation.

- The SIG(0) public keys only have to be available to the DDNS receiver.

DDNS
(but, modulo complexity, scanners would also work)

INTERNET STIFTELSEN

## A New Look At the Different Cases

Scanner + generalized notifications work

| Types of parents | Signed children | Unsigned children |
|---|---|---|
| "Registries" | OK | ? |
| Other parents | OK | OK |

An important detail is that the SIG(0) keys **do not have to be present in the parent zone**. That's an artifact of the BIND9 implementation.

- The SIG(0) public keys only have to be available to the DDNS receiver.

DDNS
(but, modulo complexity, scanners would also work)

DDNS FTW
(I see no other solution)

INTERNET
STIFTELSEN

## A New Look At the Different Cases

Scanner + generalized notifications work

I see no solutions here, "sort of lost"

| Types of parents | Signed children | Unsigned children |
|---|---|---|
| "Registries" | OK | "SOL" |
| Other parents | OK | OK |

An important detail is that the SIG(0) keys **do not have to be present in the parent zone**. That's an artifact of the BIND9 implementation.

- The SIG(0) public keys only have to be available to the DDNS receiver.

DDNS
(but, modulo complexity, scanners would also work)

DDNS FTW
(I see no other solution)

## Could All CDS/CSYNC Scanning Be Replaced By DDNS Updates?

This sounds like a technical question but it really isn't. It's a market question.

- From a technical POV DDNS updates would work fine in all cases, including parents that are registries.
- But that would require changes in the DNS market. Changes cost money. I don't think the benefits of automatic delegation synchronisation is enough to drive such changes.
- The focus here is primarily on all the zone cuts in the rest of the DNS hierarchy.

## Let's Talk About Efficiency. And Simplicity

Never mind that I don't really believe that the registries will do parent/child synching via DDNS updates.

- The drawbacks of scanners are known: **inefficient**, **slow convergence**, **require DNSSEC** and they are an **additional complex service** for the parent.

What about DDNS-based synching (regardless of the type of parent)?

- DDNS is **efficient**, **converge fast** and **does not require DNSSEC**.

But what about the complexity? Isn't a DDNS "reciever" about as complex as a scanner?

- Not really. A DDNS receiver is a rather simple piece of code, which only needs to run when a DDNS update actually arrives.
- The receiver, after evaluating the update, could easily emit a series of "add RR, delete RR, delete RR, add RR, add RR, delete RR, . . ." instructions to be executed on the next zone reload.

## Summary

| Types of parents | Signed children | Unsigned children |
|---|---|---|
| "Registries" | **NOTIFY+Scanner** | **"SOL"** |
| Other parents | **DDNS** | **DDNS** |

- Using the mechanism for locating the message target from the generalised notifications...
- and separating parent verification from parent zone update...
- combined with SIG(0)-authenticated DDNS updates...
- enables an efficient and scalable solution to the issue of automatic synchronisation of delegation information.

The method has the added advantages of being less complex than CDS/CSYNC scanners and also work independently of DNSSEC.

INTERNET STIFTELSEN

## Thanks & Contact Information

| | |
|---|---|
| **Johan Stenstam** | `johan.stenstam@internetstiftelsen.se` |
| **Working code** | `https://github.com/johanix/gen-notify-test` |

INTERNET STIFTELSEN

## How Should The SIG(0) Public Keys Be Distributed?

This is obviously an important issue.

- The child `foo.parent.` must have access to both the private and the public key with the name `foo.parent.`
- The receiver in the parent end must have access to the public key `foo.parent.`

While this problem is central, it is important to be aware that it is a **bootstrapping issue**. Once the key is known to the parent, the child is able to initiate a rollover of the SIG(0) key via a DDNS update.

- There is no difference between a DDNS update to change the NS RRset or an update to change the KEY RRset.

INTERNET STIFTELSEN

## Distribution of SIG(0) Public Keys, cont'd

There are various different alternatives:

- Provision the public key as part of the initial delegation. For non-registries this is usually a somewhat manual process.
- Add a method for uploading the public key via an authenticated web portal.
- Communicate the public key to the parent via whatever mechanism is used to communicate changes to the NS RRset or glue.

INTERNET STIFTELSEN

## Distribution of SIG(0) Public Keys, cont'd

There are various different alternatives:

- Provision the public key as part of the initial delegation. For non-registries this is usually a somewhat manual process.
- Add a method for uploading the public key via an authenticated web portal.
- Communicate the public key to the parent via whatever mechanism is used to communicate changes to the NS RRset or glue.

Honorable mention to this method:

- In a DNSSEC-signed hierarchy the parent could scan for a `KEY` record in the child zone.
  - But we don't like scanners and we want a solution also for the non-DNSSEC cases, so this is not sufficient.

INTERNET STIFTELSEN