Automating DNS Delegation Management via DDNS

Update for IETF 126

`draft-ietf-dnsop-delegation-mgmt-via-ddns-02`

Johan Stenstam Erik Bergström Leon Fernandez

The Swedish Internet Foundation

IETF 126, Vienna — July 2026

Where We Are

Adopted WG document. Last presented at IETF 125.

- -01 was presented at IETF 125 (mutual auth, KeyState, bootstrap methods).
- -02 (current) is published in the datatracker.
- Companion `draft-berra-dnsop-keystate-03` now also published — which closes the cross-reference dependency.

Where We Are

Adopted WG document. Last presented at IETF 125.

- `-01` was presented at IETF 125 (mutual auth, KeyState, bootstrap methods).
- `-02` (current) is published in the datatracker.
- Companion `draft-berra-dnsop-keystate-03` now also published — which closes the cross-reference dependency.

Today's update: what changed in `-02`, and where we think it stands.

Recap: What the Draft Specifies

A mechanism for a child to manage its delegation data in the parent via DNS UPDATE, secured with `SIG(0)`:

1. Child discovers the parent's UPDATE Receiver via `DSYNC` (RFC 9859).
2. Child sends the change (NS, DS, glue) as a plain RFC 2136 / RFC 3007 UPDATE, SIG(0)-signed (RFC 2931).
3. UPDATE Receiver discovers and validates the child's `SIG(0)` key, applies policy, updates the parent.

Recap: What the Draft Specifies

A mechanism for a child to manage its delegation data in the parent via DNS UPDATE, secured with `SIG(0)`:

1. Child discovers the parent's UPDATE Receiver via `DSYNC` (RFC 9859).
2. Child sends the change (NS, DS, glue) as a plain RFC 2136 / RFC 3007 UPDATE, SIG(0)-signed (RFC 2931).
3. UPDATE Receiver discovers and validates the child's `SIG(0)` key, applies policy, updates the parent.

The draft introduces steps 1 and 3 — one discovery problem at each end. Step 2 is plain, standard DNS UPDATE; this draft changes nothing there.

Recap: What the Draft Specifies

A mechanism for a child to manage its delegation data in the parent via DNS UPDATE, secured with `SIG(0)`:

1. Child discovers the parent's UPDATE Receiver via `DSYNC` (RFC 9859).
2. Child sends the change (NS, DS, glue) as a plain RFC 2136 / RFC 3007 UPDATE, SIG(0)-signed (RFC 2931).
3. UPDATE Receiver discovers and validates the child's `SIG(0)` key, applies policy, updates the parent.

The draft introduces steps 1 and 3 — one discovery problem at each end. Step 2 is plain, standard DNS UPDATE; this draft changes nothing there.

Works for signed *and* unsigned children; no need for scanners anywhere.

Changes in -02

- KEY changes are a distinct, bootstrap-only update class — separate from NS/glue/DS delegation updates (reconciles the receiver policy).
- RCODE/EDE made explicit: key *known but not trusted* ⇒ `REFUSED` + EDE; `BADKEY` means *unknown*.
- Name-scoping authorization rule strengthened to MUST (with a registrar-multi-child exception).
- Scanner-equivalence checks pointed at RFC 7344/8078 (DS) and RFC 7477 (CSYNC).
- UPDATE channel: TCP (or DoT) recommended — which has a consequence, see next slide.
- EDE registry gains mnemonics; IANA section complete.

–02: This Path Is Ready for Post-Quantum *Today*

PQ key and/or signature sizes is the great obstacle to PQ DNSSEC. Not on this path.

- Delegation UPDATEs are *infrequent* and go over TCP/DoT — not the per-query UDP path.
- So this path imposes no significant wire-size constraint on the SIG(0) signature.
- Large ML-DSA (FIPS 204) / SLH-DSA (FIPS 205) keys and signatures are therefore *not* a deployment obstacle here — though they remain one on ordinary DNSSEC paths.

-02: This Path Is Ready for Post-Quantum *Today*

PQ key and/or signature sizes is the great obstacle to PQ DNSSEC. Not on this path.

- Delegation UPDATEs are *infrequent* and go over TCP/DoT — not the per-query UDP path.
- So this path imposes no significant wire-size constraint on the SIG(0) signature.
- Large ML-DSA (FIPS 204) / SLH-DSA (FIPS 205) keys and signatures are therefore *not* a deployment obstacle here — though they remain one on ordinary DNSSEC paths.

There is no operational reason to prefer a classical algorithm on this path.

— *Modulo IANA algorithm code point allocation*

–02: This Path Is Ready for Post-Quantum *Today*

PQ key and/or signature sizes is the great obstacle to PQ DNSSEC. Not on this path.

- Delegation UPDATEs are *infrequent* and go over TCP/DoT — not the per-query UDP path.
- So this path imposes no significant wire-size constraint on the SIG(0) signature.
- Large ML-DSA (FIPS 204) / SLH-DSA (FIPS 205) keys and signatures are therefore *not* a deployment obstacle here — though they remain one on ordinary DNSSEC paths.

There is no operational reason to prefer a classical algorithm on this path.

— *Modulo IANA algorithm code point allocation*

It works nicely. . . Our testbed rolls the child KSK *very frequently* over exactly this path — `DSYNC`-announced UPDATE Receiver, UPDATE signed with an `MLDSA44` `SIG(0)` key.

-02: Asking Your Providers to Publish the SIG(0) Key

The `at-ns` bootstrap is the parent announcing *that it will look* for the child's `KEY` in a signed provider zone — the RFC 9615 trick, one label over:

```
_sig0key.child.parent._signal.ns.provider.example.  IN KEY ...
```

- `_signal` is RFC 9615's label (CDS/CDNSKEY bootstrap); `_sig0key` is new here, to keep the two cases apart.
- But announcing where to look does not put the key there — and the child cannot publish it itself: that name is in the *provider's* zone. It has to *ask*.

-02: Asking Your Providers to Publish the SIG(0) Key

The `at-ns` bootstrap is the parent announcing *that it will look* for the child's `KEY` in a signed provider zone — the RFC 9615 trick, one label over:

```
_sig0key.child.parent._signal.ns.provider.example.  IN KEY ...
```

- `_signal` is RFC 9615's label (CDS/CDNSKEY bootstrap); `_sig0key` is new here, to keep the two cases apart.
- But announcing where to look does not put the key there — and the child cannot publish it itself: that name is in the *provider's* zone. It has to *ask*.

-02 suggests (does not require) using the `pubkey` key of `HSYNCPARAM` (`draft-leon-dnsop-signaling-zone-owner-intent`): published in the *child* zone, so the provider knows the publication intent:

```
child.parent.  IN HSYNCPARAM ...  pubkey ...
```

Companion: draft-berra-dnsop-keystate-03

The gap it closes: the child and the parent can silently drift out of sync on the SIG(0) key (due to Murphy's Law if nothing else).

- Without KeyState, the child would find out only when an UPDATE is REFUSED — i.e. at the moment it actually needed the delegation change to go through.
- Recovery may mean re-bootstrapping a new key, possibly *manually*. The delay is unbounded.

Companion: draft-berra-dnsop-keystate-03

The gap it closes: the child and the parent can silently drift out of sync on the SIG(0) key (due to Murphy's Law if nothing else).

- Without KeyState, the child would find out only when an UPDATE is REFUSED — i.e. at the moment it actually needed the delegation change to go through.
- Recovery may mean re-bootstrapping a new key, possibly *manually*. The delay is unbounded.
- KeyState makes it checkable in advance: a new EDNS(0) option lets the child proactively *ask* the parent about a key's state.
  - No UPDATE required — just a QUERY.
  - And the parent can provide a detailed answer about the key's exact state.

Status & Next Steps

- The -02 draft is content-complete, we know of no open issues.
- There is a complete implementation: `github.com/johanix/tdns` — including experimental PQ-safe signing of the UPDATEs.

Status & Next Steps

- The -02 draft is content-complete, we know of no open issues.
- There is a complete implementation: `github.com/johanix/tdns` — including experimental PQ-safe signing of the UPDATEs.

We suggest that this document is ready for WG Last Call.

Questions?

`draft-ietf-dnsop-delegation-mgmt-via-ddns-02`

Companions:

- RFC 9859 (`DSYNC`)
- `draft-berra-dnsop-keystate-03` (now published)

Contact: `johan.stenstam@internetstiftelsen.se`