DNSOP Working Group
Internet-Draft
Intended status: Standards Track
Expires: 4 September 2025

L. Fernandez
E. Bergstr旦m
J. Stenstam
The Swedish Internet Foundation
S. Crocker
Edgemoor Research Institute

3 March 2025

Signaling Zone Owner Intent
draft-leon-dnsop-signaling-zone-owner-intent-00

Abstract

This document introduces a standardized mechanism for zone owners to signal their intent regarding DNS provider responsibilities through DNS itself. It defines a new DNS RRtype, HSYNC (Horizontal Synchronization), that enables zone owners to designate which providers are authorized to serve and/or sign their zones, control whether providers or the zone owner manages the NS RRset, and specify zone transfer chain configurations.

The HSYNC record allows DNS providers to discover each other and establish secure communication channels, either via DNS messages secured by SIG(0) signatures or via a RESTful API secured by TLS. This provider-to-provider communication via Agents enables automated coordination for tasks such as NS RRset management, zone transfers, and DNSSEC-related operations. This specification covers the provider discovery and communication establishment aspects.

The document defines two new roles to facilitate this synchronization: the Agent responsible for provider-to-provider communication and the Combiner which merges unsigned zone data from the zone owner with managed data from providers.

While a distributed DNSSEC multi-signer architecture (similar to "model 2" in RFC8901) is an important application of this framework, the HSYNC mechanism supports broader provider synchronization needs.

The specific synchronization algorithms for multi-signer operation are described in [I-D.draft-ietf-dnsop-dnssec-automation]. Specification for how to express these algorithms over provider-to-provider communication is left for a follow-up document.

TO BE REMOVED: This document is being collaborated on in Github at: https://github.com/johanix/draft-leon-dnsop-signaling-zone-owner-intent (https://github.com/johanix/draft-leon-dnsop-signaling-zone-owner-intent). 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

Copyright Notice

Table of Contents

1. Introduction

DNS zone owners often need to work with multiple DNS providers to serve their zones. These providers may have different responsibilities - some may sign the zone, some may only serve it, and some may do both. Traditionally, the configuration of these providers and their responsibilities has been handled through manual processes and provider-specific mechanisms.

This document presents a standardized mechanism for zone owners to signal their intent regarding DNS provider responsibilities through DNS itself. It defines a new DNS RRtype, HSYNC, that allows zone owners to:

* Designate which providers should serve the zone.

* Specify whether each provider should sign the zone.

* Control whether providers or the zone owner manages the NS RRset.

* Specify on a provider-level how the zone transfer chain should be setup.

* Enable providers to locate each other and establish secure communication.

By publishing this information in the DNS, zone owners ensure all providers receive consistent configuration information. This enables automated coordination between providers for tasks like:

* NS RRset management across multiple providers.

* Addition or removal of providers.

* Transition between different signing configurations.

* Management of DNSSEC-related records when multiple signers are used.

* Zone transfer chain configuration.

The intent of this document is to define a framework for secure provider-to-provider communication, based directly on intent expressed by the zone owner.

Lots of work remain with specification of the details for the various synchronization processes that will be expressed over this framework. That will take place in follow-up documents.

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

1.1. 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

Provider: In the context of this document the term "provider" always indicate a "DNS provider", i.e. an entity that provides DNS services, eg. DNSSEC-signing and/or authoritative nameservice. ...

3. Requirements

The requirements for an architecture facilitating DNS provider synchronization are defined as follows:

* Zone owners MUST be able to signal to their DNS providers information sufficient for the providers to identify each other and establish secure communication.

* All signaling from zone owner to DNS providers SHOULD be carried out via data in the served zone, ensuring that all providers receive the same configuration information at approximately the same time.

* Zone owners MUST be able to explicitly specify which DNS providers should serve and/or sign their zones.

* Zone owners MUST be able to signal the intent to onboard an additional DNS provider. This MUST automatically initiate the appropriate provider synchronization processes.

* Zone owners MUST be able to signal the intent to offboard an existing DNS provider. This MUST automatically initiate the appropriate provider synchronization processes.

* By engaging DNS providers for signing, the zone owner MUST give up control over the following records:

  - All DNSSEC-related records in the zone.

  - Any CDS and/or CSYNC RRsets.

* It SHOULD be possible but NOT MANDATORY for the zone owner to also delegate the management of the NS RRset to the set of DNS providers.

* DNS providers MUST be able to locate and establish secure communication with each other based on the information provided by the zone owner in the DNS via the HSYNC RRset.

* The architecture SHOULD support both DNS-based and API-based communication between providers.

* The architecture SHOULD allow for smooth transitions between different provider configurations without service interruption.

4. DNS Provider Synchronization Scenarios

The HSYNC framework supports a variety of scenarios where zone owners need to coordinate multiple DNS providers. The following scenarios illustrate the range of use cases this mechanism enables:

4.1. Coordinated NS Record Management

A zone owner uses two DNS providers - one signs and serves the zone while another only serves it. Through the HSYNC RRset, the zone owner signals to both providers who is authorized to do what. The providers' Agents establish secure communication channels, allowing them to coordinate NS RRset management across all authoritative nameservers without manual intervention. The zone owner can decide whether to retain control of NS records or delegate this responsibility to the providers.

4.2. Multi-Provider DNSSEC Redundancy

A zone owner needs to eliminate the "signing" single point of failure in their DNSSEC setup. By contracting with multiple "multi-signer capable" DNS providers and using data from the HSYNC RRset, the zone owner enables each provider to:

* Locate other designated providers via the HSYNC RRset and establish secure communications.

* Coordinate DNSKEY, CDS, CSYNC and NS RRset management.

* Sign the zone using its own DNSKEYs while publishing a DNSKEY RRset that includes keys from all authorized signers.

* Distribute the signed zone to its authoritative nameservers and possibly to non-signing downstream providers.

This creates a fully redundant DNSSEC infrastructure with no single point of failure.

4.3. Provider Transition Management

A zone owner wishes to replace their current DNSSEC-signing provider with a new one. Using the mechanism provided by the HSYNC RRset, they are able to:

* Add a new HSYNC record with State="ON" for the incoming provider, initiating the onboarding process.

* Allow the automated synchronization between providers to handle key exchange and transition.

* Once the new provider is fully operational, change the HSYNC State for the outgoing provider to "OFF" when convenient.

* The providers then automatically coordinate the safe removal of the outgoing provider’s data.

This entire process maintains continuous service and valid signatures while transitioning between DNS providers.

4.4. Delegated NS Management

A zone owner wants DNS providers to handle NS RRset management while retaining control of other zone data. By setting the NSMgmt field to "AGENT" in the HSYNC RRset, the zone owner explicitly delegates NS management responsibility to the DNS providers. The DNS providers then coordinate to maintain a consistent NS RRset across all authoritative servers, adding or removing nameservers as needed based on the current set of authorized providers.

4.5. Phased Migration to Multi-Provider Architecture

A zone owner currently using a single provider wants to implement a more robust architecture but prefers a gradual transition. They can:

* First add a single HSYNC record designating their current provider, making no immediate operational changes
* Later add a second HSYNC record to designate an additional provider
* Allow the providers to automatically coordinate the transition
* Optionally delegate NS management to the providers by changing NSMgmt from "OWNER" to "AGENT"

This approach enables a controlled, phased migration to a more resilient multi-provider architecture.

5. The Agent: Integrated Signer vs Separate Agent

In a distributed setup there must be a service located with each DNS Provider that manages communication with other DNS Providers. This is referred to as the Agent.

It is possible to implement support for the synchronization and communication needs directly into an existing component of the provider’s provisioning infrastructure (which may be as simple as an authoritative nameserver with or without the ability to do online DNSSEC signing). In that case this component implements the Agent functionality.

However, it is also possible to separate the synchronization and communication needs into a separate agent. This Agent sits next to the existing infrastructure, and is under the same administrative control (the "DNS Provider"), but is a separate piece of software. Each Agent is configured as a "secondary nameserver" and receives the (usually signed) zone. In this document the functional separation using a distinct Agent is used for clarity, not as a statement on preferred implementation choice.

The "separate Agent" design has the major advantage of leaving the DNSSEC-signer (if any) outside of the synchronization and communication complexity. The requirements are only that the Agent is treated as a normal secondary (it receives NOTIFY messages and is able to request zone transfers).

6. Source of Truth

A common design for DNSSEC signing (regardless of multi-signer) is to use a separate, bump-on-the-wire Signer. This is a Signer that receives the unsigned zone via an incoming zone transfer, signs the zone, and publishes the signed zone via an outbound zone transfer. In such a design the source of truth has been split up between the "zone owner" (source of truth for all non-DNSSEC zone data), and the Signer (source of truth for all DNSSEC data in the zone plus the DNSKEY RRset).

In the proposed architecture the source of truth is further split up into three participants:

* The zone owner is the source of truth for all unsigned zone data, except DNSSEC data and possibly the NS RRset.

* The Signer is the source of truth for all data generated via DNSSEC signing: own DNSKEYs, NSEC/NSEC3 RRs, RRSIGs, etc.

* The Agent is the source of truth for the RRsets that must be kept in sync across all the Signers for the zone. This includes the DNSKEYs from other providers, CDS and CSYNC RRsets. Possibly also the NS RRset.

The NS RRset is an interesting special case. Traditionally the NS RRset is maintained by the zone owner, but based on data from the DNS providers (as authoritative nameservers is a primary service for the DNS provider). However, in the proposed architecture the NS RRset should preferably be maintained by the Agents. For this reason the proposed design makes control of the NS RRset explicit and the responsibility of the zone owner to choose whether to retain control or delegate to the Agents. Hence:

* The Agent is the source of truth for the NS RRset, subject to the policy of the zone owner, as described in the HSYNC RRset.

Making the control of the NS RRset explicit is useful regardless of whether a zone uses multiple signers or single signer, as this makes the zone owner intent explicit.

To be able to keep the Signer as simple as possible, the changes to the NS, DNSKEY, CDS and CSYNC RRsets must be introduced into the unsigned zone before the zone reaches the Signer. Likewise, to keep the zone owner as simple as possible (i.e. not involved in the details of the multi-signer automation) these changes must be introduced into the unsigned zone after the zone leaves the zone owner.

6.1. The Combiner

The consequence of these requirements is that the DNSKEY, CDS and CSYNC RRsets (and possibly the NS RRset) are maintained via a separate piece of software inserted between the zone owner and the Signer. This is referred to as the Combiner.

The Combiner has the following features:

* It supports inbound zone transfer of the unsigned zone from the zone owner.

* It receives updates for the NS, DNSKEY, CDS and CSYNC RRsets from the Agent. Typically the mechanism used is DNS UPDATE with a TSIG signature, as this is easy to configure in a local context. However, other mechanisms, including APIs, are possible.

* It stores all data received from the Agent separate from the zone data received from the zone owner.

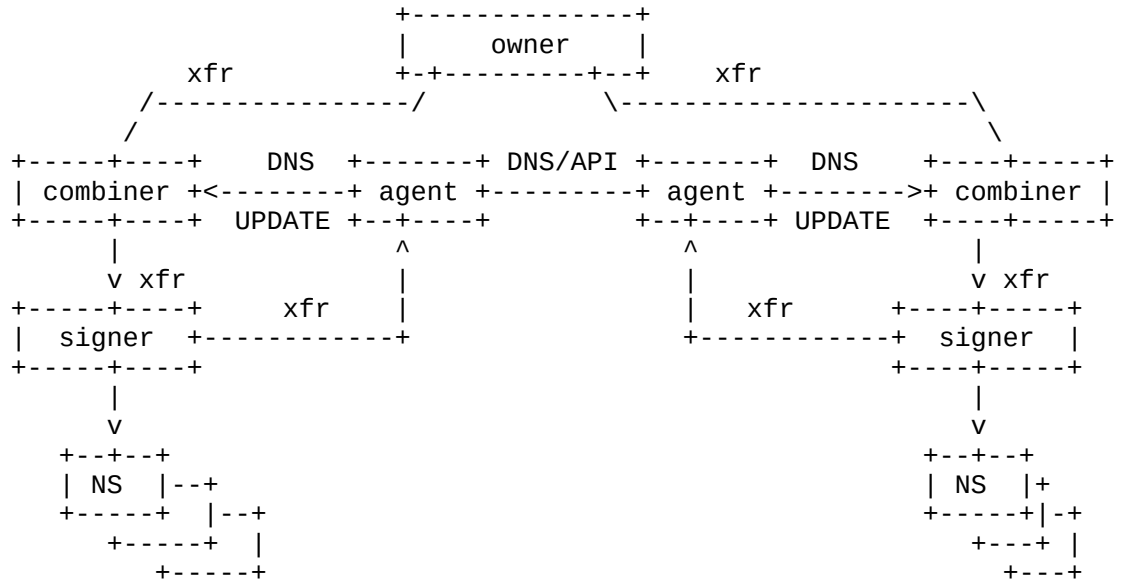
* Whenever it receives a new unsigned zone from the zone owner it COMBINES zone data from the zone owner (the majority of the zone) with specific zone data under control of the Agent: three specific RRsets, all in the apex of the zone: the DNSKEY, CDS and CSYNC RRsets. According to zone owner policy expressed in the HSYNC RRset it will also update the NS RRset.

* It is policy free (apart from being limited to the four specified RRsets). I.e. the Combiner is not making any judgement about what data to include in the zone from the four defined RRsets. That judgement is the role of the Agent.

* It does not sign the zone.

* It provides outbound zone transfer of the combined zone to the Signer.

Example setup with two signers showing the logical flow of zone data between the zone owner, the Combiner, the Signer and the Agent:



6.2. The DNS Provider

A "DNS Provider" is a term that is most commonly used to refer to an entity that provides authoritative DNS service to one or more zone owners. In the context of this document it is used to refer to an entity that provides some subset of the following services:

* Signing a zone received from the zone owner.

* Serving the zone via a set of authoritative nameservers.

* Distributing the signed zone to other downstream DNS Providers.

In addition to the above services a DNS Provider MUST also provide:

* An Agent for synchronization with other DNS Providers

* A Combiner for the management of changes to the zone via the synchronization among Agents (if it provides a signer)

I.e. in the setup above there are two DNS Providers, both of which are "complete" in the sense that they provide all three of the above services.

7. Identifying the Designated DNS Providers

It is the responsibility of the zone owner to choose a set of "DNS Providers", either internal or external to the zone owner's organization. These DNS Providers MUST be clearly and uniquely designated via publication in the HSYNC RRset, located at the apex of the zone and consisting of one HSYNC record for each signer.

The HSYNC RRset MUST be added, by the zone owner, to the, typically unsigned, zone that the zone owner maintains so that this RRset is visible to the downstream DNS Providers and their Agents.

8. The HSYNC RRset

The HSYNC RR has the zone name that publishes the HSYNC RRset as the owner name (i.e. the HSYNC RRset must be located at the apex of the zone). The RDATA consists of five fields "State","NSMgmt", "Sign", "Identity" and "Upstream":

zone.example. IN HSYNC State NSMgmt Sign Identity Upstream

State: Unsigned 8-bit. Defined values are 1=ON and 2=OFF. The value 0 is an error. Values 3-127 are presently undefined. Values 128-255 are reserved for private use. The presentation format MUST use the tokens "ON" and "OFF".

NSMgmt: Unsigned 8-bit. Defined values are 1=Zone owner and 2=Agent. The value 0 is an error. Values 3-255 are presently undefined (and not expected to be defined). The presentation format MUST use the tokens "OWNER" and "AGENT".

Sign: Unsigned 8-bit. Defined values are 1=SIGN and 2=NOSIGN. The value 0 is an error. If Sign=YES for a particular HSYNC record, then the signer associated with that Identity is a designated signer for the zone. The presentation format MUST use the tokens "SIGN" and "NOSIGN".

Identity: Domain name. Used to uniquely identify the Agent for the DNS Provider that the Agent represents.

Upstream: Domain name. Used to uniquely identify the upstream DNS Provider that this DNS Provider is a downstream of. The special case of "." is used to signal that this DNS Provider either has no upstream (is the zone owner), or that the upstream is to be configured manually.

Example:

```
zone.example.  IN HSYNC ON AGENT SIGN agent.provider.example.
upstream.
```

8.1. Semantics of the HSYNC State Field

The HSYNC State field is used to signal to all Agents what the status of each DNS Provider is from the point-of-view of the zone owner. The two possible values are "ON" and "OFF" where "ON" means that the DNS Provider is a currently a designated DNS Provider for the zone (or in the process of being on-boarded and "OFF" means that the DNS Provider is previously a designated DNS Provider for the zone that is in the process of being offboarded.

One use for the "OFF" state is that the offboarding process involves the remaining DNS Providers (hence the signalling) and it is important to know which DNS Provider is being offboarded so that the correct data may be removed in the correct order, eith during the multi-signer "remove signer" process (see [RFC8901]) or, a simpler "remove auth nameserver" process.

Once the offboarding process is complete the HSYNC RR for the offboarded DNS Provider may be removed from the zone at the zone owners discretion.

Another use for the State=OFF is during initial setup of a new DNS provider. As long as State=OFF, no data from the provider must be used by other providers. However, it is possible to verify that communication, etc, works as intended.

8.2. Semantics of the HSYNC NSMgmt Field

The NSMgmt field is used to signal to the Agents who is responsible for the contents of the NS RRset for the zone. The two possible values are "OWNER" and "AGENT".

The value "OWNER" signals that the zone owner is responsible for the NS RRset and is responsible for updating the NS RRset (either with or without the unified data from all Agents). In this case the Agents MUST NOT instruct the Combiner to update the NS RRset.

The value "AGENT" means that the Agents representing DNS Providers that sign the zone are responsible for the contents of the NS RRset. In this case the these Agents MUST instruct the local Combiner to update the NS RRset with the unified NS RRset data from all Agents.

8.2.1. Limitation of Scope for HSYNC-based NS Management

For the purpose of this document the NSMgmt Field in the HSYNC record only covers the NS RRset. I.e. it does not include the address records of in-bailiwick authoritative nameservers. The reasons are:

* Limiting the possibility of DNS Providers "polluting" the name space of the zone.

* Keeping the specification simpler, as the concept of "delegated" NS management is new.

It is possible to make an argument for delegating management of address records for in-bailiwick authoritative nameservers, but this document does not.

8.3. Semantics of the HSYNC Sign Field

The Sign field is used to signal to all Agents whether the zone owner requests that the DNS Provider that the Agent represents should sign the zone or not. The two possible values are "SIGN" and "NOSIGN" where "SIGN" means that the Agent represents a a DNS Provider that should sign the zone and "NOSIGN" means that the Agent does not.

When Sign=NOSIGN the Agent MUST still participate in the communication between Agents for the zone, to ensure proper synchronization of data between providers. If NSmgmt=AGENT, the Agent should still instruct the Combiner to update the NS RRset, otherwise no.

9. Communication Between Agents

For the communication between Agents there are two choices that need to be made among the designated Agents for a zone. The first is what "transport" to use for the communication. The second is what "synchronization" model to use when executing future synchronization processes.

The two defined transport alternatives are:

* DNS-based communication (mandatory to support)

* REST API-based communication

Each has pros and cons and at this point in time it is not clear that one always is better than the other. To simplify the choice of transport DNS-based communication is mandatory to support and the REST API-based communication may only be used if all Agents support it. Supported transports are signaled in the Provider-Synchronization EDNS(0) Option (see section NNN below).

The two defined synchronization alternatives are:

* Leader/Follower synchronization (mandatory to support)

* Peer-to-Peer synchronization

Just as for transport, supported synchronization models are signaled in the Provider-Synchronization EDNS(0) Option (see section Provider-Synchronization EDNS(0) Option (Section 9.5) below).

Regardless of the synchronization model and communication method used, the Agents SHOULD exchange all needed information about the zone and the DNS Provider they represent to enable the synchronization processes to execute correctly. This includes notifications about changes to DNSKEYs, changes to the NS RRset, etc. Depending on synchronization model it may also include instructions for changes to the zone.

In all cases the information published by a DNS provider to allow other providers to locate its Agent MUST be DNSSEC-signed.

9.1. Agent Communication via DNS

This transport alternative is based on the observation that all the communication needs between Agents can be expressed via DNS messages. Notifications are sent as DNS NOTIFY messages. Requests for changes to a zone are sent as DNS UPDATE messages, etc. One remaining communication requirement is for how to communicate information about the current state between Agents in an ongoing synchronization process. For this reason a dedicated EDNS(0) opcode specifically for provider synchronization is proposed.

This model is based on [I-D.draft-berra-dnsop-keystate] that solves a similar problem for delegation synchronization between child and parent, which has already been implemented and shown to work.

9.2. Agent Communication via REST API

REST APIs are well-known and a natural fit for many distributed systems. The challenge is mostly in the initial setup of secure communication. The certificates need to be validated, preferably without a requirement on trusting a third party CA. The API endpoints for each Agent need to be located. Once secure communication has been established, using a REST API for Agent communication is straight-forward.

9.3. Locating Remote Agents

When an Agent receives a zone via zone transfer from the Signer it will analyze the zone to see whether it contains an HSYNC RRset. If there is no HSYNC RRset the zone MUST be ignored by the Agent from the point-of-view of provider synchronization.

If, however, the zone does contain an HSYNC RRset then the Agent MUST analyze this RRset to identify the other Agents for the zone via their target names in each HSYNC record. If any of the other Agents listed in the HSYNC RRset is previously unknown to this Agent then secure communication with this other Agent MUST be established.

Secure communication can be achieved via various transports and it is up to the Agents in the zone’s HSYNC RRset to determine amongst themselves. This document proposes two transports: "DNS" and "API". "DNS" is designated as as a baseline that Agents MUST support to be compliant.

The following two subsections describe the mechanism by which an Agent SHOULD locate a remote Agent and establish secure DNS-based and API-based communications, respectively.

9.3.1. Locating a Remote DNS Transport Agent

Locating a remote Agent using the DNS mechanism consists of the following steps:

* Lookup and DNSSEC-validate a URI record for the DNS protocol for the HSYNC identity. This provides the domain name and port to which DNS messages should be sent.

* Lookup and DNSSEC-validate the SVCB record of the URI record target to get the IP addresses to use for communication with the remote Agent.

* If both the URI record and the SVCB record both include information about the target port then the port information in the SVCB MUST take precedence.

* Lookup and DNSSEC-validate the KEY record of the URI record target name. This enables verification of the SIG(0) public key of the remote Agent once communication starts.

Example: given the following HSYNC record for a remote Agent:

zone.example. IN HSYNC ON AGENT SIGN agent.provider.com. agent.zone.example.

The local Agent will look up the URI record for agent.provider.com:

_dns._tcp.agent.provider.com. IN URI 10 10 "dns://ns.provider.com:5399/" _dns._tcp.agent.provider.com. IN RRSIG URI 窶ヲ

which triggers a lookup for ns.provider.com. SVCB to get the IPv4 and IPv6 addresses as ipv4hints and ipv6hints in the response to the SVCB query:

ns.provider.com. IN SVCB 1 ipv4hint=5.6.7.8 ipv6hint=2001::53
ns.provider.com. IN RRSIG SVCB 窶ヲ

and also a look up for the KEY record for ns.provider.com, which may look like this:

ns.provider.com. IN KEY 窶ヲ ns.provider.com. IN RRSIG KEY 窶ヲ

Once all the DNS lookups and DNSSEC-validation of the returned data has been done, the local Agent is able to initiate communication with the remote Agent and verify the identity of the responding party via the validated KEY record for the remote Agents SIG(0) public key.

9.3.2. Locating a Remote API Transport Agent

Locating a remote Agent using the API mechanism consists of the following steps:

* Lookup and DNSSEC-validate the URI record for for the HTTPS protocol for the HSYNC identity. This provides the base URL that will be used to construct the individual API endpoints for the REST API. It also provides the port to use.

* Lookup and DNSSEC-validate the SVCB record for the URI record target. This provides the IP-addresses to use for communication with the Agent.

* If both the URI record and the SVCB record both include information about the target port then the port information in the SVCB MUST take precedence.

* Lookup and DNSSEC-validate the TLSA record for the port and protocol specified in the URI record. This will enable verification of the certificate of the remote Agent once communication starts.

Example: given the following HSYNC record for a remote Agent:

zone.example. IN HSYNC ON AGENT YES agent.provider.com. agent.zone.example.

the local Agent will look up the URI record for agent.provider.com:

_https._tcp.agent.provider.com. IN URI 10 10 寰徂ttps://api.provider.com:443/api/v2/寰 _https._tcp.agent.provider.com. IN RRSIG URI …

which triggers a lookup for api.provider.com IPv4 and IPv6 addresses as hints in an SVCB RR:

api.provider.com. IN SVCB 1 ipv4hint=1.2.3.4 ipv6hint=2001::bad:cafe:443 api.provider.com. IN RRSIG SVCB …

Now we know the IP-address and the port as well as the base URL to use. Finally the TLSA record for _443._tcp.api.provider.com is looked up, with a response that may look like this:

_443._tcp.api.provider.com. IN TLSA 3 1 1 ….
_443._tcp.api.provider.com. IN RRSIG TLSA …

Once all the DNS lookups and DNSSEC-validation of the returned data has been done, the local Agent is able to initiate communication with the remote Agent and verify the identity of the responding party via the TLSA record for the remote Agents certificate.

9.3.2.1. Fallback to DNS-based Communication

If the API-based communication fails, either because needed DNS records are missing, the TLSA record fails to validate the remote Agents certificate or the remote Agent simply doesn’t respond, the local Agent MUST fall back to DNS-based communication.

9.4. The Initial HELLO Phase

When two Agents need to communicate with each other for the first time (because they are both designated DNS providers for the same zone), they need to establish secure communication. This is done in a "HELLO" phase where the Agents exchange information about their capabilities.

If all the information needed for API-based transport for the remote party was available, the Agent SHOULD attempt an API-based HELLO. If, however, this fails for some reason, it should fall back to DNS-based HELLO.

9.4.1. DNS-based HELLO Phase

When using DNS-based communication the HELLO phase is done by sending a NOTIFY(SOA) for the zone that triggered the need for communication. The NOTIFY message MUST contain a Provider-Synchronization EDNS(0) Option (see section Provider Synchronization EDNS(0) Option (Section 9.5) below).

In the Provider-Synchronization EDNS(0) Option the OPERATION field MUST have the value "HELLO" (1). Furthermore, the Agent signals its transport and synchronization capabilities in the TRANSPORT and SYNCHRONIZATION fields. This message is signed with the SIG(0) key for the local Agent for which the public key is published as a KEY record for the Agent.

In the response to the NOTIFY, the remote Agent does the same and the two Agents can now verify each other’s identity and are also aware of the other Agents transport and synchronization capabilities.

9.4.2. API-based HELLO Phase

When using API-based communication the HELLO phase is done by sending a REST API POST request to the remote Agent at the "/hello" endpoint. The request MUST contain a JSON encoded object with the following fields:

* "transport": The transport capabilities of the local Agent.

* "synchronization": The synchronization capabilities of the local Agent.

The response MUST contain a JSON object with the following fields:

* "transport": The transport capabilities of the remote Agent.

* "synchronization": The synchronization capabilities of the remote Agent.

9.4.3. Interpretation of the HELLO Responses

Once an Agent has received HELLO responses from all other Agents that are designated signers for the zone, it knows the capabilities of the Agents as a group. It can then use this information to determine which transport to use:

* If all Agents support API-based communication, the Agents will use API-based communication for this zone.

* If one or more Agents only support DNS-based communication, the Agents will use DNS-based communication for this zone.

Likewise, each Agent now knows the provider synchronization capabilities of the other Agents and can determine which synchronization model to use:

* If all Agents support the Peer-to-Peer synchronization model, the Agents will use the Peer-to-Peer synchronization model for this zone.

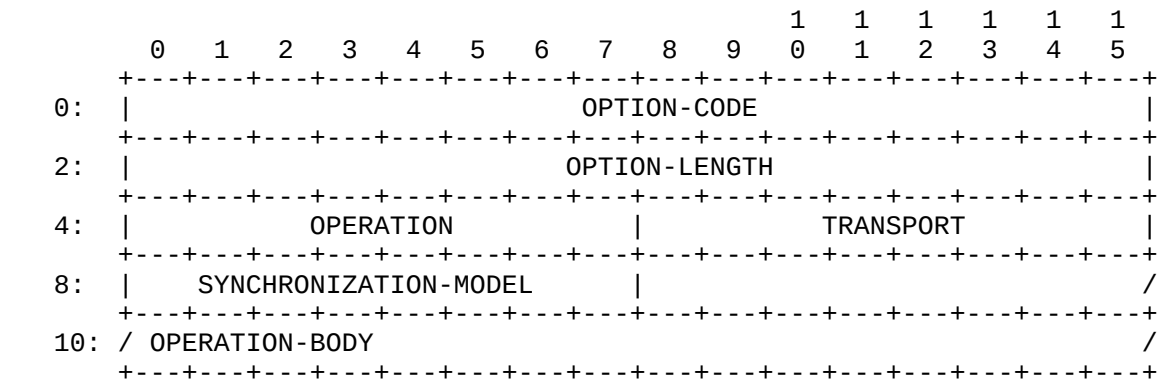
* If one or more Agents only support the Leader/Follower synchronization model, the Agents will use the Leader/Follower synchronization model for this zone.

9.5. Provider-Synchronization EDNS(0) Option Format

This document uses an Extended Mechanism for DNS (EDNS0) [RFC6891] option to include DNS Provider synchronization information in DNS messages.

This option is structured the same way as the KeyState option described in [I-D.draft-berra-dnsop-keystate], which has been implemented and shown to work for a similar use case. The requirements for DNS Provider synchronization are sufficiently different that it is not possible to re-use the KeyState OPT also for this purpose and therefore a new EDNS(0) option is defined here.

The Provider-Synchronization EDNS(0) option is structured as follows:



Field definition details:

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

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 OPERATION-BODY field (which may be zero octets long).

OPERATION: 8 bits. Signals the type of operation the message performs. This document defines the two operations HELLO and HEARTBEAT. For a complete distributed multi-signer specification a number of additional operations will need to be allocated to be able to describe the states in the different multi-signer processes. This allocation must be done either in a revision to this document or in a subsequent document.

TRANSPORT: 8 bits. Encodes the transport capabilities of the Agent. With 8 bits it is possible to define up to 8 different transports of which this document defines two: DNS and API.

SYNCHRONIZATION-MODEL: 8 bits. Encodes the synchronization capabilities of the Agent. With 8 bits it is possible to define up to 8 different synchronization models of which this document identifies two: Leader/Follower and Peer-to-Peer.

OPERATION-BODY: Variable-length. Used to carry operation-specific parameters.

9.5.1. Encoding Transport Capabilities in the Provider-Synchronization EDNS(0) Option

An Agent signals the union of its transport capabilities by setting the corresponding bits to 1.

0: DNS transport supported (baseline, MUST be supported by all Agents)

1: API transport supported

2-7: unused

9.5.2. Encoding Synchronization Capabilities in the Provider-Synchronization EDNS(0) Option

An Agent signals its synchronization capabilities by setting the corresponding bits to 1.

0: Leader/Follower multi-signer synchronization supported

1: Peer-to-Peer multi-signer synchronization supported

2-7: unused

10. Sequence Diagram Example of Establishing Secure Comms - "The Hello Phase"

The procedure of locating another Agent and establishing a secure communication, referred to as "The Hello Phase" is examplified in the sequence diagram below.

The procedure is as follows:

1. The Agents receive a zone via zone transfer. By analyzing the HSYNC RRset each Agent become aware of the identities of the other Agents for the zone. I.e. each Agent knows which other Agents it needs to communicate with. Communication with each of these, previously unknown, remote Agents is referred to as "NEEDED".

2. Each Agent starts aquiring the information needed to establish secure communications with any previously unknown Agents. Here we only illustrate the baseline case where DNS-based communications is to be used in the following phase. Once all needed information has been collected the communication with this remote Agent is considered to be "KNOWN".

3. Once an Agent has received the required information (URI, SVCB and KEY records in the baseline case) it sends a NOTIFY message with a dedicated Provider-Synchronization OPT code with OPERATION="HELLO". The sender uses this OPT field to signal its transport and synchronization capabilities. Similarly, the responder signals its capabilities using the same field.

4. When an Agent either gets a NOERROR response to its NOTIFY OPT(hello) message or responds with a NOERROR, it transitions out of "The Hello Phase" with the exchanging party and they transition to the next phaste where they start sending NOTIFY OPT(heartbeat) signals instead. The communication with the remote Agent is now considered to be in the "OPERATIONAL" state.

In the case where one Agent is aware of the need to communicate with another Agent, but the other is not (eg. the zone transfer was delayed for one of them), the slower one SHOULD respond with a RCODE=REFUSED to any NOTIFY OPT(hello) it receives. Once it is ready, it will send its own NOTIFY OPT(hello) which should be responded to with a RCODE=NOERROR.

11. Responsibilities of an Agent

Each Agent has certain responsibilites, depending on supported transports methods.

11.1. Enabling Remote Agents to Locate This Agent

For a group of Agents to be able to communicate securely and synchronize data for a zone, each Agent must ensure that the DNS records needed for secure communication with other Agents are published:

* URI, SVCB and KEY records required for DNS-based communication secured by SIG(0).

* URI, SVCB and TLSA records required for API-based communication secured by TLS (if supported).

* All of the above MUST be published in a DNSSEC-signed zone under the domain name that is the identity of the Agent.

11.2. Enabling Remote Agents to Lookup Zone Data Added By This Agent

When using DNS transport between Agents, four types of information is needed to be conveyed from one party to another:

1. Notifications (sent as DNS NOTIFY).

2. Retrieval of existing data (looked up via DNS QUERY).

3. Changes to existing data (sent as DNS UPDATE).

In addition there is also a need for the Agents to signal state information to each other. One obvious case of this is when the Agents need to signal the state of an ongoing synchronization process to each other:

1. Provider "state" information (sent via the Provider-Synchronization EDNS(0) OPT).

The second case, i.e. looking up data for a zone that is particular to a specific DNS Provider is typically about the DNSKEY records added by that signer or the NS records representing the authoritative nameservers for that DNS Provider. This is looked up under domain names constructed from the name of the served zone and the identity of the DNS Provider.

For each zone that is managed, the Agent must ensure that the data needed for synchronization with other Agents is published:

* The DNSKEY RRset for the zone consisting of the DNSKEYs that the local Signer for this DNS Provider uses to sign the zone.

* The CDS RRset for the zone, representing the KSK that the local Signer uses to sign the zone (when needed).

* The NS RRs for the zone, consisting of the NS records of the authoritative nameservers that this DNS Provider is responsible for.

* All of the above MUST be published in a DNSSEC-signed zone under the domain name that is the concatenation of the zone name and the identity of the Agent. Example for the zone "zone.example" and the Agent "agent.provider":

```
zone.example.agent.provider.  IN DNSKEY ...
zone.example.agent.provider.  IN RRSIG DNSKEY ...
zone.example.agent.provider.  IN NS ... zone.example.agent.provider.
IN RRSIG NS ...
```

12. Migration from Single-Signer to Multi-Signer

The migration from a single-signer to a multi-signer architecture is done by changing from only having a single designated signer in the HSYNC RRset to having multiple designated signers (i.e. the SIGN field changed from "NOSIGN" to "SIGN"). This may be done in several steps.

12.1. Adding a single HSYNC record to an already signed zone

Adding a single HSYNC record to a zone that is already signed by the DNS provider "provider.com" with NSmgmt=OWNER is a no-op that does not change anything in the zone:

```
zone.example.  IN HSYNC ON OWNER SIGN agent.provider.com. upstream.
```

The zone was already signed by the DNS provider "provider.com" and the provider added any needed DNSSEC records, including DNSKEYs. The zone NS RRset was managed by the zone owner. All of this is unchanged by the addition of the HSYNC RRset.

What does change is possibly the zone transfer chain, if the specified upstream is different from previously.

12.2. Changing the HSYNC NSMGMT Field from AGENT To OWNER

Each Agent publishes the data it contributes to the zone under the domain name {zone}.{identity}. I.e. the zone DNSKEYs that the Agent agent.provider.com. uses are published as:

```
zone.example.agent.provider.com.  DNSKEY ...
zone.example.agent.provider.com.  DNSKEY ...
```

Likewise, the NS records for the zone are published as:

```
zone.example.ns.agent.provider.com.  NS ...
zone.example.ns.agent.provider.com.  NS ...
```

To migrate from "owner maintained" NS RRset to "Agent maintained", the zone owner must verify that the NS RRset as published by the Agent is correct and in sync with the NS RRset as published by the zone owner itself. After this verification the zone owner changes the HSYNC NSmgmt field in the existing HSYNC records from NSmgmt=OWNER to NSmgmt=AGENT.

12.3. Migrating from a Multi-Signer Architecture Back to Single-Signer.

If, for some reason, a zone owner wants to migrate back to a single-signer architecture (i.e. offboarding the second DNS Provider), the process is essentially the reverse of the migration from single-signer to multi-signer:

1. The zone owner offboards the second signing DNS Provider (only keeping one signing DNS Provider).

When offboarding the second signing DNS Provider is signalled via the HSYNC RRset, the multi-step process "remove signer" (as defined in [I-D.draft-ietf-dnsop-dnssec-automation]) is initiated to remove the second DNS Provider from the zone in a series of steps.

The zone is now essentially back to a single-signer architecture. Once the offboarding is complete, the zone owner may remove the HSYNC RRset designating the offboarded DNS Provider from the zone.

TO BE REMOVED BEFORE PUBLICATION: # Rationale

12.4. Choice of the HSYNC Mnemonic

Initially the mnemonic "MSIGNER" was used for the HSYNC RRset. However, as work progressed it became clear that we want also non-signing DNS Providers to be able to participate. So the RRset is a signalling mechanism from zone owner to DNS Providers, some of which may or may not be instructed to sign the zone. Therefore we suggest the mnemonic "HSYNC" to indicate that this is a mechanism for "horizontal synchronization" inside a zone.

But the mnemonic chosen is a very minor point and should a better suggestion come up it would be great.

12.5. Separation of Agent and Combiner

It is possible to integrate all three multi-signer components (Signer, Agent and Combiner) into a single piece of software (or two pieces, depending on the preferred way of slicing the functionality). However, such a composite module would be a fairly complex piece of software.

This document aims to describe the functional separation of the different components rather than make a judgement on software design alternatives. Hence possible implementation choices are left to the implementer.

13. Security Considerations

An architecture for automated multi-provider zone management is a complex system with a number of components. The authors believe that the only way to make such an architecture useful in practice is via automation. However, automation is a double-edged sword. It can both make the system more robust and more vulnerable.

While all communication between Agents is authenticated (either via SIG(0) signatures or TLS), the signalling from the zone owner to the Agents is via the HSYNC RRset in an unsigned zone. This is a potential attack vector. However, securing zone transfers from zone owner to DNS providers is a well-known issue with lots of existing solutions (TSIG, zone transfer via a secure channel, zone transfer-over-TLS, etc). Employing some of these solutions is strongly recommended.

From a vulnerability point-of-view this architecture introduces several new components into the zone signing and publication process. In particular the Combiner and the Agents are new components that need to be secure. The Combiner has the advantage of not having to announce its location to the outside world, as it only needs to communicate with internal components (the zone owner, the Signer and the Agent).

The Agent is more vulnerable. It needs to be discoverable by other Agents and hence it is also discoverable by an adversary. On the other hand, the Agents are not needed for a new zone to be signed and published, they are only needed when there are changes that require the Agents to synchronize, which is an infrequent event.

Furthermore, should an Agent be unable to fulfill its role during the execution of a change requiring synchronization, whether it is a complex multi-signer process or perhaps only a change to the NS RRset, the synchronization process will simply stop where it is. Regardless of where the stop (or rather pause) occurs, the zone will be fully functional (as in available and properly signed). Once the Agent is able to resume its role, the synchronization process will continue from where it left off.

14. IANA Considerations.

Note to the RFC Editor: In this section, please replace occurrences of "(This document)" with a proper reference.

14.1. HSYNC RR Type

IANA is requested to update the "Resource Record (RR) TYPEs" registry under the "Domain Name System (DNS) Parameters" registry group as follows:

Type HSYNC

Value TBD

Meaning Zone owner designation of DNS providers enabling mutual discovery

Reference (This document)

14.2. New Provider-Synchronization EDNS Option

This document defines a new EDNS(0) option, entitled "Provider-Synchronization", assigned a value of TBD "DNS EDNS0 Option Codes (OPT)" registry

TO BE REMOVED UPON PUBLICATION: https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-11 (foo)

Value	Name	Status	Reference
TBD	Provider-Synchronization	Standard	(This document)

14.3. A New Registry for EDNS Option Provider-Synchronization Operation Codes

The Provider-Synchronization option also defines an 8-bit operation field, for which IANA is requested to create and mainain a new registry entitled "Provider-Synchronization Operations", used by the Provider-Synchronization option. Initial values for the "Provider-Synchronization Operations" registry are given below; future assignments in in the 3-127 range are to be made through Specification Required review [BCP26].

OPERATION	Mnemonic	Reference
0	forbidden	(This document)
1	HELLO	(This document)
2	HEARTBEAT	(This document)
3-127	Unassigned	(This document)
128-255	Private Use	(This document)

15. References

15.1. 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/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>.

[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/info/rfc2931>.

[RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <https://www.rfc-editor.org/info/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/info/rfc6891>.

[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>.

[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>.

15.2. 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.draft-berra-dnsop-keystate] Bergstrm, E., Fernandez, L., and J. Stenstam, "Signalling Key State Via DNS EDNS(0) OPT", Work in Progress, Internet-Draft, draft-berra-dnsop-keystate-01, 7 February 2025, <https://datatracker.ietf.org/doc/html/draft-berra-dnsop-keystate-01>.

[I-D.draft-ietf-dnsop-dnssec-automation] Wisser, U., Huque, S., and J. Stenstam, "DNSSEC automation", Work in Progress, Internet-Draft, draft-ietf-dnsop-dnssec-automation-03, 19 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-dnssec-automation-03>.

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

* This document is derived from the earlier document draft-leon-dnsop-distributed-multi-signer-00

  Initial public draft.

Authors' Addresses

Leon Fernandez
The Swedish Internet Foundation
Sweden
Email: leon.fernandez@internetstiftelsen.se

Erik Bergstrm
The Swedish Internet Foundation
Sweden

Email: erik.bergstrom@internetstiftelsen.se

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

Steve Crocker
Edgemoor Research Institute
United States
Email: steve@shinkuro.com