Workgroup: DNSOP Working Group
Internet-Draft:
draft-ietf-dnsop-dns-catalog-zones-05
Published: 7 March 2022
Intended Status: Standards Track
Expires: 8 September 2022
Authors: P. van Dijk L. Peltan
PowerDNS CZ.NIC
O. Sury W. Toorop
Internet Systems Consortium NLnet Labs
K. Monshouwer
P. Thomassen
deSEC, SSE - Secure Systems Engineering

DNS Catalog Zones

Abstract

This document describes a method for automatic DNS zone provisioning among DNS primary and secondary nameservers by storing and transferring the catalog of zones to be provisioned as one or more regular DNS zones.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The content of a DNS zone is synchronized amongst its primary and secondary nameservers using AXFR and IXFR. However, the list of zones served by the primary (called a catalog in [RFC1035]) is not automatically synchronized with the secondaries. To add or remove a zone, the administrator of a DNS nameserver farm not only has to add or remove the zone from the primary, they must also add/remove the zone from all secondaries, either manually or via an external application. This can be both inconvenient and error-prone; it is also dependent on the nameserver implementation.

This document describes a method in which the catalog is represented as a regular DNS zone (called a "catalog zone" here), and transferred using DNS zone transfers. As zones are added to or removed from the catalog zone, these changes are distributed to the secondary nameservers in the normal way. The secondary nameservers then add/remove/modify the zones they serve in accordance with the changes to the catalog zone. Other use-cases of nameserver remote configuration by catalog zones are possible, where the catalog consumer might not be a secondary.

2. Terminology

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.

Catalog zone A DNS zone containing a DNS catalog, that is, a list of DNS zones and associated properties.

Member zone A DNS zone whose configuration is published inside a catalog zone.

Member node The DNS name in the Catalog zone representing a Member zone.

$CATZ Used in examples as a placeholder to represent the domain name of the catalog zone itself. $OLDCATZ and $NEWCATZ are used to discuss migration a member zone from one catalog zone $OLDCATZ to another catalog zone $NEWCATZ.

Catalog producer An entity that generates and is responsible for the contents of the catalog zone.

Catalog consumer An entity that extracts information from the catalog zone (such as a DNS server that configures itself according to the catalog zone's contents).

3. Description

A catalog zone is a DNS zone whose contents are specially crafted. Its records primarily constitute a list of PTR records referencing other DNS zones (so-called "member zones"). The catalog zone may contain other records indicating additional metadata (so-called "properties") associated with these member zones.

Catalog consumers SHOULD ignore any RR in the catalog zone which is meaningless or useless to the implementation.

Authoritative servers may be preconfigured with multiple catalog zones, each associated with a different set of configurations.

Although the contents of a catalog zone are interpreted and acted upon by nameservers, a catalog zone is a regular DNS zone and so must adhere to the standards for such zones.

A catalog zone is primarily intended for the management of a farm of authoritative nameservers. The content of catalog zones may not be accessible from any recursive nameserver.

4. Catalog Zone Structure

4.1. SOA and NS Records

As with any other DNS zone, a catalog zone MUST have a syntactically correct SOA record and at least one NS record at its apex.

The SOA record's SERIAL, REFRESH, RETRY and EXPIRE fields [RFC1035] are used during zone transfer. A catalog zone's SOA SERIAL field MUST increase when an update is made to the catalog zone's contents as per serial number arithmetic defined in [RFC1982]. Otherwise, catalog consumers might not notice updates to the catalog zone's contents.

There is no requirement to be able to query the catalog zone via recursive nameservers. Catalog consumers MUST ignore and MUST NOT assume or require NS records at the apex. However, at least one is still required so that catalog zones are syntactically correct DNS zones. A single NS RR with a NSDNAME field containing the absolute name "invalid." is RECOMMENDED [RFC2606].

4.2. Member Zones

The list of member zones is specified as a collection of member nodes, represented by domain names under the owner name "zones" where "zones" is a direct child domain of the catalog zone.

The names of member zones are represented on the RDATA side (instead of as a part of owner names) of a PTR record, so that all valid domain names may be represented regardless of their length [RFC1035]. This PTR record MUST be the only record in the PTR RRset with the same name. More than one record in the RRset denotes a broken catalog zone which MUST NOT be processed (see Section 5.1).

For example, if a catalog zone lists three zones "example.com.", "example.net." and "example.org.", the member node RRs would appear as follows:

```
<unique-1>.zones.$CATZ 0 IN PTR example.com.
<unique-2>.zones.$CATZ 0 IN PTR example.net.
<unique-3>.zones.$CATZ 0 IN PTR example.org.
```

where <unique-N> is a label that tags each record in the collection. <unique-N> has an unique value in the collection.

Member node labels carry no informational meaning beyond labeling member zones. A changed label may indicate that the state for a zone needs to be reset (see Section 5.6).

Having the zones uniquely tagged with the <unique-N> label ensures that additional RRs can be added below the member node (see Section 4.4).

The CLASS field of every RR in a catalog zone MUST be IN (1).

The TTL field's value is not defined by this memo. Catalog zones are for authoritative nameserver management only and are not intended for general querying via recursive resolvers.

4.3. Global Properties

Apart from catalog zone metadata stored at the apex (NS, SOA and the like), catalog zone information is stored in the form of "properties". Catalog consumers SHOULD ignore properties they do not understand.

This specification defines a number of so-called properties, as well as a mechanism to allow implementers to store additional information in the catalog zone with Custom properties, see Section 4.5. The meaning of such custom properties is determined by the implementation in question.

Some properties are defined at the global level; others are scoped to apply only to a specific member zone. This document defines a single mandatory global property in Section 4.3.1. Member-specific properties are described in Section 4.4.

More properties may be defined in future documents.

4.3.1. Schema Version (version property)

The catalog zone schema version is specified by an integer value embedded in a TXT RR named version.$CATZ. All catalog zones MUST have a TXT RRset named version.$CATZ with exactly one RR. Catalog consumers MUST NOT apply catalog zone processing to zones without the expected value in the version.$CATZ TXT RR, but they may be transferred as ordinary zones. For this memo, the value of the version.CATZ TXT RR MUST be set to "2", i.e.:

```
version.$CATZ 0 IN TXT "2"
```

NB: Version 1 was used in a draft version of this memo and reflected the implementation first found in BIND 9.11.

4.4. Member Zone Properties

Each member zone MAY have one or more additional properties, described in this chapter. These properties are completely optional and catalog consumers SHOULD ignore those it does not understand. Member zone properties are represented by RRsets below the corresponding member node.

4.4.1. Change of Ownership (coo property)

The coo property facilitates controlled migration of a member zone from one catalog to another.

A Change Of Ownership is signaled by the coo property in the catalog zone currently "owning" the zone. The name of the new catalog is in the value of a PTR record in the old catalog. For example if member "example.com." will migrate from catalog zone $OLDCATZ to catalog zone $NEWCATZ, this appears in the $OLDCATZ catalog zone as follows:

```
<unique-N>.zones.$OLDCATZ 0 IN PTR example.com.
coo.<unique-N>.zones.$OLDCATZ 0 IN PTR $NEWCATZ
```

The PTR RRset MUST consist of a single PTR record. More than one record in the RRset denotes a broken catalog zone which MUST NOT be processed (see Section 5.1).

When a consumer of catalog zone $OLDCATZ receives an update which adds or changes a coo property for a member zone in $OLDCATZ signalling a new owner $NEWCATZ, it does *not* migrate the member zone immediately.

This is because the catalog consumer may not have the <unique-N> identifier associated with the member zone in $NEWCATZ and because name servers do not index Resource Records by RDATA, it may not know whether or not the member zone is configured in $NEWCATZ at all. It may have to wait for an update of $NEWCATZ adding or changing that member zone. When a consumer of catalog zone $NEWCATZ receives an update of $NEWCATZ which adds or changes a member zone, *and* that consumer had the member zone associated with $OLDCATZ, *and* there is a coo property of the member zone in $OLDCATZ pointing to $NEWCATZ, *only then* it will reconfigure the member zone with the for $NEWCATZ preconfigured settings.

Unless the member node label (i.e. <unique-N>) for the member is the same in $NEWCATZ, all associated state for a just migrated zone MUST be reset (see Section 5.6). Note that the owner of $OLDCATZ allows for the zone associated state to be taken over by the owner of

$NEWCATZ by default. To prevent the takeover, the owner of $OLDCATZ has to enforce a zone state reset by changing the member node label (see Section 5.6) before or simultaneous with adding the coo property. (see also Section 7)

The old owner may remove the member zone containing the coo property from $OLDCATZ once it has been established that all its consumers have processed the Change of Ownership.

4.4.2. Groups (group property)

With a group property, consumer(s) can be signalled to treat some member zones within the catalog zone differently.

The consumer MAY apply different configuration options when processing member zones, based on the value of the group property. The exact handling of configuration referred to by the group property value is left to the consumer's implementation and configuration. The property is defined by a TXT record in the sub-node labelled group.

The producer MAY assign a group property to all, some, or none of the member zones within a catalog zone. The producer MUST NOT assign more than one group property to one member zone.

The consumer MUST ignore either all or none of the group properties in a catalog zone.

The value of the TXT record MUST be at most 255 octets long and MUST NOT contain whitespace characters. The consumer MUST interpret the value case-sensitively.

4.4.2.1. Example

```
<unique-1>.zones.$CATZ        0 IN PTR    example.com.
group.<unique-1>.zones.$CATZ  0 IN TXT    sign-with-nsec3
<unique-2>.zones.$CATZ        0 IN PTR    example.net.
group.<unique-2>.zones.$CATZ  0 IN TXT    nodnssec
```

In this case, the consumer might be implemented and configured in the way that the member zones with "nodnssec" group assigned will not be signed with DNSSEC, and the zones with "sign-with-nsec3" group assigned will be signed with DNSSEC with NSEC3 chain.

By generating the catalog zone (snippet) above, the producer signals how the consumer shall treat DNSSEC for the zones example.net. and example.com., respectively.

4.5. Custom Properties (*.ext properties)

Implementations and operators of catalog zones may choose to provide their own properties. Custom properties can occur both globally, or for a specific member zone. To prevent a name clash with future properties, such properties should be represented below the label ext.

ext is not a placeholder, so a custom property would have domains names as follows:

```
<your-property>.ext.$CATZ                     # for a global custom proper
<your-property>.ext.<unique-N>.zones.$CATZ  # for a member zone custom p
```

<your-property> may consist of one or more labels.

Implementations MAY use such properties on the member zone level to store additional information about member zones, for example to flag them for specific treatment (such as ...).

Further, implementations MAY use custom properties on the global level to store additional information about the catalog zone itself. While there may be many use cases for this, a plausible one is to store default values for custom properties on the global level, then overriding them using a property of the same name on the member level (= under the ext label of the member node) if so desired. A property description should clearly say what semantics apply, and whether a property is global, member, or both.

The meaning of the custom properties described in this section is determined by the implementation alone, without expectation of interoperability. A catalog consumer SHOULD ignore custom properties it does not understand.

5. Nameserver Behavior

5.1. General Requirements

As it is a regular DNS zone, a catalog zone can be transferred using DNS zone transfers among nameservers.

Although they are regular DNS zones, catalog zones contain only information for the management of a set of authoritative nameservers. For this reason, operators may want to limit the systems able to query these zones. It may be inconvenient to serve some contents of catalog zones via DNS queries anyway due to the nature of their representation. A separate method of querying entries inside the catalog zone may be made available by nameserver implementations (see Section 6).

Catalog updates should be automatic, i.e., when a nameserver that supports catalog zones completes a zone transfer for a catalog zone, it SHOULD apply changes to the catalog within the running nameserver automatically without any manual intervention.

As with regular zones, primary and secondary nameservers for a catalog zone may be operated by different administrators. The secondary nameservers may be configured as catalog consumer to synchronize catalog zones from the primary, but the primary's administrators may not have any administrative access to the secondaries.

Nameservers MAY allow loading and transfer of broken zones with incorrect catalog zone syntax (as they are treated as regular zones), but catalog consumers MUST NOT process such broken zones as catalog zones. For the purpose of catalog processing, the broken catalogs MUST be ignored.

5.2. Member zone name clash

If there is a clash between an existing zone's name (either from an existing member zone or otherwise configured zone) and an incoming member zone's name (via transfer or update), the new instance of the zone MUST be ignored and an error SHOULD be logged.

A clash between an existing member zone's name and an incoming member zone's name (via transfer or update), may be an attempt to migrate a zone to a different catalog, but should not be treated as one except as described in {#cooproperty}.

5.3. Member zone removal

When a member zone is removed from a specific catalog zone, an authoritative server MUST NOT remove the zone and associated state data if the zone was not configured from that specific catalog zone. Only when the zone was configured from a specific catalog zone, and the zone is removed as a member from that specific catalog zone, the zone and associated state (such as zone data and DNSSEC keys) MUST be removed.

5.4. Member node name change

When via a single update or transfer, the member node's label value (<unique-N>) changes, catalog consumers MUST process this as a member zone removal including all the zone's associated state (as described in Section 5.3), immediately followed by processing the member as a newly to be configured zone in the same catalog.

5.5. Migrating member zones between catalogs

If all consumers of the catalog zones involved support the coo property, it is RECOMMENDED to perform migration of a member zone by following the procedure described in Section 4.4.1. Otherwise a migration of member zone from a catalog zone $OLDCATZ to a catalog zone $NEWCATZ has to be done by: first removing the member zone from $OLDCATZ; second adding the member zone to $NEWCATZ.

If in the process of a migration some consumers of the involved catalog zones did not catch the removal of the member zone from $OLDCATZ yet (because of a lost packet or down time or otherwise), but did already see the update of $NEWCATZ, they may consider the update adding the member zone in $NEWCATZ to be a name clash (see Section 5.2) and as a consequence the member is not migrated to $NEWCATZ. This possibility needs to be anticipated with a member zone migration. Recovery from such a situation is out of the scope of this document. It may for example entail a manually forced retransfer of $NEWCATZ to consumers after they have been detected to have received and processed the removal of the member zone from $OLDCATZ.

5.6. Zone associated state reset

It may be desirable to reset state (such as zone data and DNSSEC keys) associated with a member zone.

A zone state reset may be performed by a change of the member node's name (see Section 5.4).

6. Implementation Notes

Catalog zones on secondary nameservers would have to be setup manually, perhaps as static configuration, similar to how ordinary DNS zones are configured. The secondary additionally needs to be configured as a catalog consumer for the catalog zone to enable processing of the member zones in the catalog, such as automatic synchronized of the member zones for secondary service.

An administrator may want to look at data inside a catalog zone. Typical queries might include dumping the list of member zones, dumping a member zone's effective configuration, querying a specific property value of a member zone, etc. Because of the structure of catalog zones, it may not be possible to perform these queries intuitively, or in some cases, at all, using DNS QUERY. For example, it is not possible to enumerate the contents of a multi-valued property (such as the list of member zones) with a single QUERY. Implementations are therefore advised to provide a tool that uses either the output of AXFR or an out-of-band method to perform queries on catalog zones.

7. Security Considerations

As catalog zones are transmitted using DNS zone transfers, it is RECOMMENDED that catalog zone transfer are protected from unexpected modifications by way of authentication, for example by using TSIG [RFC8945], or Strict or Mutual TLS authentication with DNS Zone transfer over TLS [RFC9103].

Use of DNS UPDATE [RFC2136] to modify the content of catalog zones SHOULD similarly be authenticated.

Zone transfers of member zones SHOULD similarly be authenticated. TSIG shared secrets used for member zones SHOULD NOT be mentioned in the catalog zone data. However, key identifiers may be shared within catalog zones.

Catalog zones reveal the zones served by the consumers of the catalog zone. It is RECOMMENDED to limit the systems able to query these zones. It is RECOMMENDED to transfer catalog zones confidentially [RFC9103].

Administrative control over what zones are served from the configured name servers shifts completely from the server operator (consumer) to the "owner" (producer) of the catalog zone content.

With migration of member zones between catalogs using the coo property, it is possible for the owner of the target catalog (i.e. $NEWCATZ) to take over all associated state with the zone from the original owner (i.e. $OLDCATZ) by maintaining the same member node label (i.e. <unique-N>). To prevent the takeover of the zone associated state, the original owner has to enforce a zone state reset by changing the member node label (see Section 5.6) before or simultaneously with adding the coo property.

8. Acknowledgements

Our deepest thanks and appreciation go to Stephen Morris, Ray Bellis and Witold Krecicki who initiated this draft and did the bulk of the work.

Catalog zones originated as the chosen method among various proposals that were evaluated at ISC for easy zone management. The chosen method of storing the catalog as a regular DNS zone was proposed by Stephen Morris.

The initial authors discovered that Paul Vixie's earlier [Metazones] proposal implemented a similar approach and reviewed it. Catalog zones borrows some syntax ideas from Metazones, as both share this scheme of representing the catalog as a regular DNS zone.

Thanks to Leo Vandewoestijne. Leo's presentation in the DNS devroom at the FOSDEM'20 [FOSDEM20] was one of the motivations to take up and continue the effort of standardizing catalog zones.

Thanks to Brian Conry, Klaus Darilion, Brian Dickson, Tony Finch, Evan Hunt, Shane Kerr, Patrik Lundin, Victoria Risk, Petr Spacek and Carsten Strotmann for reviewing draft proposals and offering comments and suggestions.

9. 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>.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

[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>.

[RFC2606] Eastlake 3rd, D. and A. Panitz, "Reserved Top Level DNS Names", BCP 32, RFC 2606, DOI 10.17487/RFC2606, June 1999, <https://www.rfc-editor.org/info/rfc2606>.

[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>.

[RFC8945] Dupont, F., Morris, S., Vixie, P., Eastlake 3rd, D., Gudmundsson, O., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", STD 93, RFC 8945, DOI 10.17487/RFC8945, November 2020, <https://www.rfc-editor.org/info/rfc8945>.

[RFC9103] Toorop, W., Dickinson, S., Sahib, S., Aras, P., and A. Mankin, "DNS Zone Transfer over TLS", RFC 9103, DOI 10.17487/RFC9103, August 2021, <https://www.rfc-editor.org/info/rfc9103>.

10. Informative References

[FOSDEM20]
Vandewoestijne, L., "Extending Catalog zones - another approach in automating maintenance", 2020, <https://archive.fosdem.org/2020/schedule/event/dns_catz/>.

[Metazones] Vixie, P., "Federated Domain Name Service Using DNS Metazones", 2005, <http://family.redbarn.org/~vixie/mz.pdf>.

Appendix A. Implementation Status

Note to the RFC Editor: please remove this entire section before publication.

In the following implementation status descriptions, "DNS Catalog Zones" refers to DNS Catalog Zones as described in this document.

* Knot DNS 3.1 (released August 2, 2021) supports full producing and consuming of catalog zones, including the group property.

* PowerDNS has a proof of concept external program called PowerCATZ, that can process DNS Catalog Zones.

* Proof of concept python scripts that can be used for both generating and consuming DNS Catalog Zones with NSD have been developed during the hackathon at the IETF-109.

Interoperability between the above implementations has been tested during the hackathon at the IETF-109.

Appendix B. Change History (to be removed before final publication)

* draft-muks-dnsop-dns-catalog-zones-00

Initial public draft.

* draft-muks-dnsop-dns-catalog-zones-01

Added Witold, Ray as authors. Fixed typos, consistency issues. Fixed references. Updated Area. Removed newly introduced custom RR TYPEs. Changed schema version to 1. Changed TSIG requirement from MUST to SHOULD. Removed restrictive language about use of DNS QUERY. When zones are introduced into a catalog zone, a primary SHOULD first make the new zones available for transfers first (instead of MUST). Updated examples, esp. use IPv6 in examples per Fred Baker. Add catalog zone example.

* draft-muks-dnsop-dns-catalog-zones-02

Addressed some review comments by Patrik Lundin.

*draft-muks-dnsop-dns-catalog-zones-03

Revision bump.

*draft-muks-dnsop-dns-catalog-zones-04

Reordering of sections into more logical order. Separation of multi-valued properties into their own category.

*draft-toorop-dnsop-dns-catalog-zones-00

New authors to pickup the editor pen on this draft

Remove data type definitions for zone properties Removing configuration of member zones through zone properties altogether

Remove Open issues and discussion Appendix, which was about zone options (including primary/secondary relationships) only.

*draft-toorop-dnsop-dns-catalog-zones-01

Added a new section "The Serial Property", introducing a new mechanism which can help with disseminating zones from the primary to the secondary nameservers in a timely fashion more reliably.

Three different ways to provide a "serial" property with a member zone are offered to or the workgroup for discussion.

Added a new section "Implementation Status", listing production ready, upcoming and Proof of Concept implementations, and reporting on interoperability of the different implementations.

*draft-toorop-dnsop-dns-catalog-zones-02

Adding the coo property for zone migration in a controlled fashion

Adding the group property for reconfigure settings of member zones in an atomic update

Adding the epoch property to reset zone associated state in a controlled fashion

*draft-toorop-dnsop-dns-catalog-zones-03

Big cleanup!

Introducing the terms catalog consumer and catalog producer

Reorganized topics to create a more coherent whole

Properties all have consistent format now

Try to assume the least possible from implementations w.r.t.:

1) Predictability of the <unique-N> IDs of member zones

2) Whether or not fallback catalog zones can be found for a member

3) Whether or not a catalog consumer can maintain state

*draft-toorop-dnsop-dns-catalog-zones-04

Move Implementation status to appendix

Miscellaneous textual improvements

coo property points to $NEWCATZ (and not zones.$NEWCATZ)

Remove suggestion to increase serial and remove member zone from $OLDCATZ after migration

More consistent usage of the terms catalog consumer and catalog producer throughout the document

Better (safer) description of resetting refresh timers of member zones with the serial property

Removing a member MUST remove zone associated state

Make authentication requirements a bit less prescriptive in security considerations

Updated implementation status for KnotDNS

Describe member node name changes and update "Zone associated state reset" to use that as the mechanism for it.

Add Peter Thomassen as co-author

Complete removal of the epoch property. We consider consumer optimizations with predictable member node labels (for example based on a hash) out of the scope of this document.

Miscellaneous editorial improvements

*draft-toorop-dnsop-dns-catalog-zones-05

Add Kees Monshouwer as co-author

Removed the "serial" property

> Allow custom properties on the global level

Authors' Addresses

Peter van Dijk
PowerDNS
Den Haag
Netherlands

Email: peter.van.dijk@powerdns.com

Libor Peltan
CZ.NIC
Czechia

Email: libor.peltan@nic.cz

Ondrej Sury
Internet Systems Consortium
Czechia

Email: ondrej@isc.org

Willem Toorop
NLnet Labs
Science Park 400
1098 XH Amsterdam
Netherlands

Email: willem@nlnetlabs.nl

Kees Monshouwer
Netherlands

Email: mind@monshouwer.eu

Peter Thomassen
deSEC, SSE - Secure Systems Engineering
Berlin
Germany

Email: peter@desec.io