
Stream: Internet Engineering Task Force (IETF)
RFC: [9432](#)
Category: Standards Track
Published: July 2023
ISSN: 2070-1721
Authors: P. van Dijk L. Peltan O. Sury W. Toorop
PowerDNS CZ.NIC Internet Systems Consortium NLnet Labs
C.R. Monshouwer P. Thomassen A. Sargsyan
deSEC, SSE - Secure Systems Engineering Internet Systems Consortium

RFC 9432

DNS Catalog Zones

Abstract

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

Status of This Memo

This is an Internet Standards Track document.

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

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <https://www.rfc-editor.org/info/rfc9432>.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction
2. Terminology
3. Description
4. Catalog Zone Structure
 - 4.1. Member Zones
 - 4.2. Properties
 - 4.2.1. Schema Version (version property)
 - 4.3. Member Zone Properties
 - 4.3.1. Change of Ownership (coo property)
 - 4.3.2. Groups (group property)
 - 4.4. Custom Properties (*.ext properties)
5. Name Server Behavior
 - 5.1. General Requirements
 - 5.2. Member Zone Name Clash
 - 5.3. Member Zone Removal
 - 5.4. Member Node Name Change
 - 5.5. Migrating Member Zones between Catalogs
 - 5.6. Zone-Associated State Reset
6. Implementation and Operational Notes
7. Security Considerations
8. IANA Considerations
9. References
 - 9.1. Normative References
 - 9.2. Informative References
- Appendix A. Catalog Zone Example
- Acknowledgements
- Authors' Addresses

1. Introduction

The content of a DNS zone is synchronized among its primary and secondary name servers using Authoritative Transfer (AXFR) and Incremental Zone Transfer (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 name server farm has to not only add or remove the zone from the primary but must also add or remove configuration for the zone from all secondaries. This can be both inconvenient and error-prone. In addition, the steps required are dependent on the name server implementation.

This document describes a method in which the list of zones is represented as a regular DNS zone (called a "catalog zone" here) and transferred using DNS zone transfers. When entries are added to or removed from the catalog zone, it is distributed to the secondary name servers just like any other zone. Secondary name servers can then add, remove, or modify the zones they serve in accordance with the changes to the catalog zone. Other use cases of name server 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, which is a list of DNS zones and associated properties.

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

Member node: A 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 of 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).

This document makes use of terminology for transfer mechanisms (AXFR and IXFR), record types (SOA, NS, and PTR), and other technical terms (such as RDATA) that are specific to the DNS. Since these terms have specific meanings in the DNS, they are not expanded upon first use in this document. For definitions of these and other terms, see [RFC8499](#).

3. Description

A catalog zone is a DNS zone whose contents are specially crafted. Its resource records (RRs) 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 **MUST** ignore any RRs in the catalog zone for which no processing is specified or which are otherwise not supported by the implementation.

Authoritative servers may be pre-configured 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 name servers, a catalog zone is a regular DNS zone and must adhere to the standards for DNS zones.

A catalog zone is primarily intended for the management of a farm of authoritative name servers and should not be expected to be accessible from any recursive name server.

4. Catalog Zone Structure

A catalog zone **MUST** follow the usual rules for DNS zones. In particular, SOA and NS record sets **MUST** be present and adhere to standard requirements (such as [RFC1982]).

Although catalog zones are not intended to be queried via recursive resolution (see [Section 7](#)), at least one NS RR is still required so that a catalog zone is a syntactically correct DNS zone. A single NS RR with a NSDNAME field containing the absolute name "invalid." is **RECOMMENDED** [RFC2606] [RFC6761].

4.1. 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 of a PTR record (instead of being represented as a part of owner names) 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. The presence of more than one record in the RRset indicates a broken catalog zone that **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 and has a unique value. When different `<unique-N>` labels hold the same PTR value (i.e., point to the same member zone), the catalog zone is broken and **MUST NOT** be processed (see [Section 5.1](#)).

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 4.2](#)).

The CLASS field of every RR in a catalog zone **MUST** be IN (1). The TTL field's value has no meaning in this context and **SHOULD** be ignored.

4.2. Properties

Catalog zone information is stored in the form of "properties".

Properties are identified by their name, which is used as an owner name prefix for one or more record sets underneath a member node (or underneath the catalog zone apex), with RR type(s) as appropriate for the respective property.

Known properties that have the correct RR type but are for some reason invalid (for example, because of an impossible value or because of an illegal number of RRs in the RRset) denote a broken catalog zone, which **MUST NOT** be processed (see [Section 5.1](#)).

This document includes a set of initial properties that can be extended via the IANA registry defined and created in [Section 8](#). Some properties are defined at the global level; others are scoped to apply only to a specific member zone. This document defines a mandatory global property in [Section 4.2.1](#). The "zones" label from [Section 4.1](#) can also be seen as a global property and is listed as such in the IANA registry in [Section 8](#). Member-specific properties are described in [Section 4.3](#).

Implementers may store additional information in the catalog zone with custom properties; see [Section 4.4](#). The meaning of such custom properties is determined by the implementation in question.

4.2.1. 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 `version` property
- zones with a `version` property with more than one RR in the RRset
- zones with a `version` property without an expected value in the `version.$CATZ TXT RR`
- zones with a `version` property with a schema version value that is not implemented by the consumer (e.g., version "1")

These conditions signify a broken catalog zone, which **MUST NOT** be processed (see [Section 5.1](#)).

For this memo, the value of the `version.$CATZ TXT RR` **MUST** be set to "2"; that is:

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

Note that Version 1 was used in an earlier draft version of this memo and reflected the implementation first found in BIND 9.11.

4.3. Member Zone Properties

Each member zone **MAY** have one or more additional properties that are described in this section. The member properties described in this document are all optional, and implementations **MAY** choose to implement all, some, or none of them. Member zone properties are represented by RRsets below the corresponding member node.

4.3.1. 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 the value of a PTR record in the relevant `coo` property in the old catalog. For example, if member "example.com." migrates from catalog zone `$OLDCATZ` to catalog zone `$NEWCATZ`, this will appear 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. The presence of more than one record in the RRset indicates a broken catalog zone, which **MUST NOT** be processed (see [Section 5.1](#)).

When a consumer of a catalog zone `$OLDCATZ` receives an update that adds or changes a `coo` property for a member zone in `$OLDCATZ`, it does *not* migrate the member zone immediately. The migration has to wait for an update of `$NEWCATZ` in which the member zone is present. Before the actual migration, the consumer **MUST** verify that the `coo` property pointing to `$NEWCATZ` is still present in `$OLDCATZ`.

Unless the member node label (i.e., <unique-N>) for the member is the same in \$NEWCATZ, all its 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 of the zone-associated state, the owner of \$OLDCATZ must remove this state by updating the associated properties or by performing a zone state reset (see [Section 5.6](#)) before or simultaneous with adding the `coo` property (see [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.

4.3.2. Groups (group property)

With a `group` property, a consumer(s) can be signaled 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. A `group` property value is stored as the entire RDATA of a TXT record directly below the member node. The exact handling of the `group` property value is left to the consumer's implementation and configuration.

The producer **MAY** assign a `group` property to all, some, or none of the member zones within a catalog zone. The producer **MAY** assign more than one `group` property to one member zone. This will make it possible to transfer group information for different consumer operators in a single catalog zone. Implementations **MAY** facilitate mapping of a specific `group` value to a specific configuration configurable *on a per catalog zone basis* to allow for producers that publish their catalog zone at multiple consumer operators. Consumer operators **SHOULD** namespace their `group` values to reduce the risk of having to resolve clashes.

The consumer **MUST** ignore `group` values it does not understand. When a consumer encounters multiple `group` values for a single member zone, it **MAY** choose to process all, some, or none of them. This is left to the implementation.

4.3.2.1. Example

`group` properties are represented by TXT RRs. The record content has no pre-defined meaning. Their interpretation is purely a matter of agreement between the producer and the consumer(s) of the catalog.

For example, the "foo" group could be agreed to indicate that a zone not be signed with DNSSEC. Conversely, an agreement could define that `group` names starting with "operator-" indicate in which way a given DNS operator should set up certain aspects of the member zone's DNSSEC configuration.

Assuming that the catalog producer and consumer(s) have established such agreements, consider the following catalog zone (snippet) that signals to a consumer(s) how to treat DNSSEC for the zones "example.net." and "example.com.":

```
<unique-1>.zones.$CATZ      0 IN PTR    example.com.
group.<unique-1>.zones.$CATZ 0 IN TXT    "foo"
<unique-2>.zones.$CATZ      0 IN PTR    example.net.
group.<unique-2>.zones.$CATZ 0 IN TXT    "operator-x-foo"
group.<unique-2>.zones.$CATZ 0 IN TXT    "operator-y" "bar"
```

In this scenario, a consumer(s) shall, by agreement, not sign the member zone "example.com." with DNSSEC. For "example.net.", the consumers, at two different operators, will configure the member zone to be signed with a specific combination of settings. The group value designated to indicate this combination of settings is prearranged with each operator ("operator-x-foo" vs. "operator-y" "bar").

4.4. Custom Properties (*.ext properties)

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

"ext" is not a placeholder. A custom property is named as follows:

```
; a global custom property:
<property-prefix>.ext.$CATZ

; a member zone custom property:
<property-prefix>.ext.<unique-N>.zones.$CATZ
```

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

Implementations **SHOULD** namespace their custom properties to limit risk of clashes with other implementations of catalog zones. This can be achieved by using two labels as the <property-prefix> so that the name of the implementation is included in the prefix: <some-setting>.<implementation-name>.ext.\$CATZ.

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

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 override 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 agreement between producer and consumer should clearly define 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.

5. Name Server Behavior

5.1. General Requirements

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

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

Name servers **MAY** allow loading and transfer of broken zones with incorrect catalog zone syntax (as they are treated as regular zones). The reason a catalog zone is considered broken **SHOULD** be communicated clearly to the operator (e.g., through a log message).

When a previously correct catalog zone becomes a broken catalog zone, it loses its catalog meaning because of an update through an incremental transfer or otherwise. No special processing occurs. Member zones previously configured by this catalog **MUST NOT** be removed or reconfigured in any way.

If a name server restarts with a broken catalog zone, the broken catalog **SHOULD NOT** prevent the name server from starting up and serving the member zones in the last valid version of the catalog zone.

Processing of a broken catalog **SHALL** start (or resume) when the catalog turns into a correct catalog zone, e.g., by an additional update (through zone transfer or updates) fixing the catalog zone.

Similarly, when a catalog zone expires, it loses its catalog meaning and **MUST** no longer be processed as such. No special processing occurs until the zone becomes fresh again.

5.2. Member Zone Name Clash

If there is a clash between an existing zone's name (from either an existing member zone or an 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 it should not be treated as one except as described in [Section 4.3.1](#).

5.3. Member Zone Removal

When a member zone is removed from a specific catalog zone, a consumer **MUST NOT** remove the zone and associated state data if the zone was not configured from that specific catalog zone. The zone and associated state (such as zone data and DNSSEC keys) **MUST** be removed from the

consumer when and only when the zone was configured initially from the same catalog. Consumer operators may consider temporarily archiving associated state to facilitate mistake recovery.

5.4. Member Node Name Change

When the member node's label value (<unique-N>) changes via a single update or transfer, catalog consumers **MUST** process this as a member zone removal, including the removal of all the zone's associated state (as described in [Section 5.3](#)), and then immediately process the member as a newly added zone to be configured 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 4.3.1](#). Otherwise, the migration of a member zone from a catalog zone `$OLDCATZ` to a catalog zone `$NEWCATZ` has to be done by first removing the member zone from `$OLDCATZ` and then adding the member zone to `$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 `$OLDCATZ` yet (because of a lost packet or downtime or otherwise) but already saw the update of `$NEWCATZ` containing the addition of that member zone, they may consider this update to be a name clash (see [Section 5.2](#)) and, as a consequence, the member is not migrated to `$NEWCATZ`. This possibility needs to be anticipated with a member zone migration. Recovery from such a situation is out of the scope of this document. For example, it may entail a manually forced retransfer of `$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 and Operational Notes

Although any valid domain name can be used for the catalog name `$CATZ`, a catalog producer **MUST NOT** use names that are not under the control of the catalog producer (with the exception of reserved names). It is **RECOMMENDED** to use either a domain name owned by the catalog producer or a domain name under a suitable name such as "invalid." [[RFC6761](#)].

Catalog zones on secondary name servers would have to be set up manually, perhaps as static configuration, similar to how ordinary DNS zones are configured when catalog zones or another automatic configuration mechanism are not in place. Additionally, the secondary 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 synchronization of the member zones for secondary service.

Operators of catalog consumers should note that secondary name servers may receive DNS NOTIFY messages [RFC1996] for zones before they are seen as newly added member zones to the catalog from which that secondary is provisioned.

Although they are regular DNS zones, catalog zones only contain information for the management of a set of authoritative name servers. To prevent unintended exposure to other parties, operators **SHOULD** limit the systems able to query these zones.

Querying/serving catalog zone contents may be inconvenient via DNS due to the nature of their representation. Therefore, an administrator may want to use a different method for looking at data inside the 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 multivalued 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.

Great power comes with great responsibility. Catalog zones simplify zone provisioning by orchestrating zones on secondary name servers from a single data source: the catalog. Hence, the catalog producer has great power and changes must be treated carefully. For example, if the catalog is generated by some script and this script generates an empty catalog, millions of member zones may get deleted from their secondaries within seconds, and all the affected domains may be offline in a blink of an eye.

7. Security Considerations

As catalog zones are transmitted using DNS zone transfers, it is **RECOMMENDED** that catalog zone transfers be protected from unexpected modifications by way of authentication, e.g., by using a Transaction Signature (TSIG) [RFC8945] or Strict or Mutual TLS authentication with DNS zone transfer over TLS or QUIC [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 their consumers, including their properties. To prevent unintentional exposure of catalog zone contents, it is **RECOMMENDED** to limit the systems able to query them and to conduct catalog zone transfers confidentially [RFC9103].

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

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. To prevent unintended provisioning of zones, a consumer(s) **SHOULD** scope the set of admissible member zones by any means deemed suitable (such as statically via regular expressions, or dynamically by verifying against another database before accepting a member zone).

With migration of member zones between catalogs using the `coo` property, it is possible for the owner of the target catalog (i.e., `$NEWCATZ`) to take over all its associated state with the zone from the original owner (i.e., `$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. IANA Considerations

IANA has created the "DNS Catalog Zones Properties" registry under the "Domain Name System (DNS) Parameters" registry as follows:

Registry Name: DNS Catalog Zones Properties

Assignment Policy: Expert Review, except for property prefixes ending in the label "ext", which are for Private Use [\[RFC8126\]](#).

Reference: RFC 9432

Note: This registry applies to Catalog Zones schema version "2" as specified in RFC 9432.

| Property Prefix | Description | Status | Reference |
|-----------------|----------------------|-----------------|-----------|
| zones | List of member zones | Standards Track | RFC 9432 |
| version | Schema version | Standards Track | RFC 9432 |
| coo | Change of Ownership | Standards Track | RFC 9432 |
| group | Group | Standards Track | RFC 9432 |
| *.ext | Custom properties | Private Use | RFC 9432 |

Table 1: DNS Catalog Zones Properties Registry

The meanings of the fields are as follows:

Property prefix: One or more domain name labels.

Description: A human-readable short description or name for the property.

Status: IETF Stream RFC status or "External" if not documented in an IETF Stream RFC.

Reference: A stable reference to the document in which this property is defined.

9. References

9.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <<https://www.rfc-editor.org/info/rfc1982>>.
- [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>>.
- [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>>.
- [RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <<https://www.rfc-editor.org/info/rfc6761>>.
- [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>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [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>>.

9.2. Informative References

- [FOSDEM20] Vandewoestijne, L., "Extending Catalog zones - another approach in automating maintenance", February 2020, <https://archive.fosdem.org/2020/schedule/event/dns_catz/>.
- [Metazones] Vixie, P., "Federated Domain Name Service Using DNS Metazones", DOI 10.1093/ietcom/e89-b.4.1144, April 2006, <<https://www.semanticscholar.org/paper/Federated-Domain-Name-Service-Using-DNS-Metazones-Vixie/dc12b0116332f5c236b05c71bbe20499f3c6c4b6>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.

Appendix A. Catalog Zone Example

The following is a full example of a catalog zone containing three member zones with various properties:

```
catalog.invalid.                                0 SOA   invalid. (
                                invalid. 1625079950 3600 600 2147483646 0 )
catalog.invalid.                                0 NS    invalid.
example.vendor.ext.catalog.invalid.             0 CNAME example.net.
version.catalog.invalid.                       0 TXT   "2"
nj2xg5b.zones.catalog.invalid.                0 PTR   example.com.
nvxxezj.zones.catalog.invalid.                0 PTR   example.net.
group.nvxxezj.zones.catalog.invalid.           0 TXT   (
                                "operator-x-foo" )
nfwxa33.zones.catalog.invalid.                 0 PTR   example.org.
coo.nfwxa33.zones.catalog.invalid.             0 PTR   (
                                newcatz.invalid. )
group.nfwxa33.zones.catalog.invalid.           0 TXT   (
                                "operator-y-bar" )
metrics.vendor.ext.nfwxa33.zones.catalog.invalid. 0 CNAME (
                                collector.example.net. )
```

Acknowledgements

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

Catalog zones originated as the chosen method among various proposals that were evaluated at Internet Systems Consortium (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 they reviewed it. Catalog zones borrow 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 FOSDEM'20 [[FOSDEM20](#)] was one of the motivations to take up and continue the effort of standardizing catalog zones.

Thanks to Joe Abley, David Blacka, Brian Conry, Klaus Darilion, Brian Dickson, Tony Finch, Evan Hunt, Shane Kerr, Warren Kumari, Patrik Lundin, Matthijs Mekking, Victoria Risk, Josh Soref, Petr Spacek, Michael StJohns, Carsten Strotmann, and Tim Wicinski for reviewing earlier draft versions and offering comments and suggestions.

Authors' Addresses

Peter van Dijk

PowerDNS
Den Haag
Netherlands
Email: peter.van.dijk@powerdns.com

Libor Peltan

CZ.NIC
Czech Republic
Email: libor.peltan@nic.cz

Ondrej Sury

Internet Systems Consortium
Czech Republic
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

Aram Sargsyan

Internet Systems Consortium

Email: aram@isc.org