DNSOP
Internet-Draft
Intended status: Standards Track
Expires: January 15, 2014

W. Hardaker
Parsons, Inc.

July 14, 2013

Child To Parent Synchronization in DNS
draft-hardaker-dnsop-csync-01

Abstract

This document specifies how a child zone in the DNS can publish a record that can indicate that a parental agent may copy and process certain records from the child zone. The existence and change of the record may be monitored by a parental agent to either assist in transferring or automatically transfer data from the child to the parent.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 15, 2014.

Table of Contents

1. Introduction

[Up front note: this document is very rough in wording. It is offered as a starting point to see if there is interest in pursuing the concepts contained herein before significant work is done on wording refinements]

This document specifies how a child zone in the DNS can publish a record that can indicate that a parental agent may copy and process certain records from the child zone. The existence and change of the record may be monitored by a parental agent to either assist in transferring or automatically transfer data from the child to the parent.

Some resource records (RRs) in a parent zone are typically expected to be in-sync with the data in the child's zone. The most common records that should match are the nameserver (NS) records and any necessary associated address (A and AAAA) "glue" records. Additionally, the children frequently have DNSKEY records which require corresponding DS record(s) in the parent. These records are referred to as "delegation records".

It has been traditionally challenging for children to keep delegation records within a parent's delegation record set up to date. This difficult has usually derived from simple operator laziness or the complexities of maintaining a large number of DNS zones. Having an automated mechanism to update a parent's set of delegation records will greatly ease the child zone operator's maintenance burden and improve the robustness of the DNS as a whole.

This draft introduces a new RR type (RRType) named "CSYNC" that indicates which delegation records within a child should be processed into the parent's DNS zone data.

1.1. Terminology Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document is aimed at the case where there is an organizational separation of the child and parent. In this case there are many different operating situations. A common case is the Registrant/ Registrar/Registry relationship. In this case the parent consists of Registrar and Registry, with different rules on what each can do or not do. To remain operating model neutral we will use the neutral word "Parental Agent" as the entity that uses results of DNS queries to inject delegation changes into the parent zone. The entity that

inserts the changes in the the DNS is called "DNS Publisher".

2. Definition of the CSYNC RRType

The CSYNC RRType contains, in its RDATA component, these parts: an SOA serial number, a set of flags and a simple bit-list indicating the DNS RRTypes in the child that should be processed by the parent agent to modify the DNS delegation records for the child within the parent's zone. Children wanting a parental agent to perform a synchronization MUST publish a CSYNC record at the apex of the child zone. Parent agent implementations MAY choose to query child zones for this record and process the data indicated by the Type Bit Map field in the RDATA of the CSYNC record. How the data is processed is later described in Section Section 2.2.

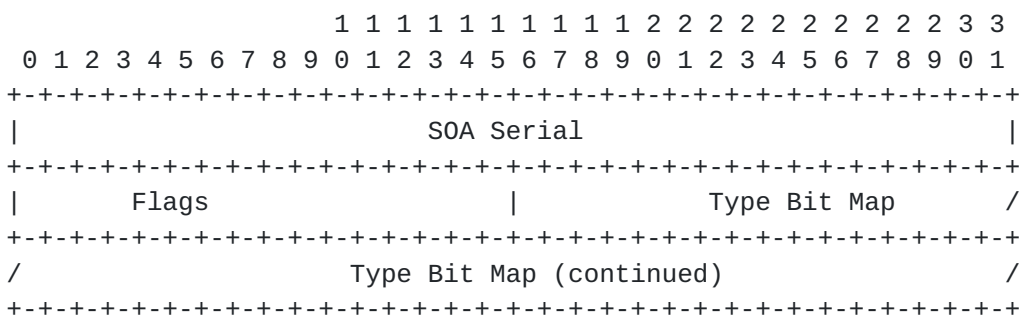
Parental agents MUST process the entire set of child data requested (i.e., all record types indicated along with all of the necessary records to support processing of that type) or else parent agents MUST NOT process any data records at all. Errors due to unsupported Type Bit Map bits or otherwise nonpunishable data SHALL result in no change to the parent zone's delegation information for the child. Parental agents SHOULD ignore a child's CSYNC RDATA set if multiple CSYNC resource records are found; only a single CSYNC record should be expected.

The parental agent MUST perform DNSSEC validation of the CSYNC RRType data and MUST perform DNSSEC validation of any data to be copied from the child to the parent. Parents MUST not process any data if any of the validation results indicate any anything other than "Secure" [RFC4034].

2.1. The CSYNC Resource Record Format

2.1.1. The CSYNC Resource Record Wire Format

The CSYNC RDATA consists of the following fields:



2.1.1.1. The SOA Serial Field

The SOA Serial field contains a copy of the 32-bit SOA serial number from the child zone. If the value is non-zero, parental agent querying children's authoritative servers MUST NOT act on data from zones advertising an SOA serial number less than this value. A special value of 0 indicates that no such restriction is in place.

Note that a child zone's current SOA serial number maybe greater than the number contained in the CSYNC record. A child SHOULD update the SOA Serial field in the CSYNC record every time the data being referenced by the CSYNC record is changed (e.g. an NS record or associated address record is changed). A child MAY choose to update the SOA Serial field to always match the current SOA serial field.

Parental agents MAY cache SOA serial numbers from data they use and refuse to process data from zones older than the last instance they pulled data from.

2.1.1.2. The Flags Field

The Flags field contains 16 bits of flags defining operations that affect the processing of the CSYNC record. The flags defined in this document are as follows:

0x00 0x01: "immediate"

The definitions for how the flags are to be used can be found later in Section Section 2.2.

The remaining flags are reserved for use by future specifications. Undefined flags MUST be set to 0 by CSYNC publishers. Parental agents MUST NOT process a CSYNC record if it contains a 1 value for a flag that is unknown to or unsupported by the parental agent.

2.1.1.2.1. The Type Bit Map Field

The Type Bit Map field indicates the record types to be processed by the parental agent, according to the procedures in Section Section 2.2. The Type Bit Map field is encoded in the same way as the Type Bit Maps field of the NSEC record, described in [RFC4034], Section 4.1.2. If a bit has been set that a parental agent implementation does not understand, the parental agent MUST NOT act upon the data. Specifically, a parental agent must not copy data blindly; A specification must exist [insert long debate here over level of specification required; IMHO: proposed IETF standard since a bit can only be used once] that defines how the data should be processed for a given bit.

2.1.2. The CSYNC Presentation Format

The CSYNC presentation format is as follows:

The SOA Serial field is represented as an integer.

The Flags field is represented as an integer.

The Type Bit Map field is represented as a sequence of RR type mnemonics. When the mnemonic is not known, the TYPE representation described in [RFC3597], Section 5, MUST be used.

2.1.3. CSYNC RR Example

The following CSYNC RR shows an example entry for "example.com" that indicates the NS, A and AAAA bits are set and should be processed by the parental agent for example.com. The parental agent should pull data only from a zone using a minimum SOA serial number of 66 (0x42 in hexadecimal).

example.com. 3600 IN CSYNC 66 1 A NS AAAA

The RDATA component of the example CSYNC RR would be encoded on the wire as follows:

```
0x00 0x00 0x00 0x42               (SOA Serial)
0x00 0x01                         (Flags)
0x00 0x04 0x60 0x00 0x00 0x08     (Type Bit Map)
```

2.2. CSYNC Data Processing

The CSYNC data must be processed as an "all or nothing" type operation. If a parental agent fails to query for any of the required records from the child, the whole operation MUST be aborted. (Note that a query resulting in "no records exist" as proven by NSEC or NSEC3 is to be considered successful).

Parental agents MAY:

Process the CSYNC record immediately after noticing it if the "immediate" flag is set. If the "immediate" flag is not set, the parental agent MUST not act until the zone administrator approves the operation through an out-of-band mechanism (such as through a web interface).

Require that the child zone administrator approve the operation through an out-of-band mechanism (such as through a web interface). I.e., a parental agent MAY choose not to support the

"immediate" flag.

Note: how the approval is done out-of-band is outside the scope of this document and likely specific to a particular parental agent.

2.2.1. Processing Procedure

The following shows a sequence of steps that SHOULD be used when collecting and processing CSYNC records from a child zone. Because DNS queries are not allowed to contain more than one question at a time, a sequence of requests will be needed. To ensure a single host is being addressed, DNS over TCP SHOULD be used to avoid conversing with multiple nodes at an anycast address.

1. Query for the child zone's SOA record

2. Query for the child zone's CSYNC record

3. Query for the child zone's data records, as required by the CSYNC record's Type Bit Map field

4. Query for the child zone's SOA record again

If the SOA records from the first and last steps have different serial numbers, then this CSYNC record set MUST NOT be processed.

If the SOA serial number(s) are less than the CSYNC record's SOA Serial Field, the record MUST NOT be processed. If state is being kept and the SOA serial number is less than the last time a CSYNC record was processed, this CSYNC record SHOULD NOT be processed.

If DNSSEC fails to validate all of the data returned as "secure", this CSYNC record MUST NOT be processed.

See the "Operational Consideration" section (Section Section 2.3) for additional guidance about processing.

2.2.2. CSYNC Record Types

This document defines how the following record types may be processed if the CSYNC Type Bit Map field indicates they should be processed.

2.2.2.1. The NS type

The NS type flag indicates that the NS records from the child zone should be copied into the parent's delegation information records for the child.

NS records found within the child's zone should be copied verbatim and the result published within the parent zone should be a matching set of NS records. Note: if NS records in the parent's delegation records for the child contain records that have been removed in the child's NS set, then they should be removed in the parent's set as well.

Parental agents MAY refuse to perform NS updates if the replacement records fail to meet NS record policies required by the parent zone (e.g. "every child zone must have at least 2 NS records").

2.2.2.2. The A and AAAA types

The A and AAAA type flags indicates that the A and AAAA, respectively, address glue records for in-bailiwick NS records within the child zone should be copied into the parent's delegation information.

Queries should be sent by the parental agent to determine the A and AAAA record addresses for each NS record within a NS set for the child that are in-bailiwick.

Note: only the matching types should be queried. E.g., if the AAAA bit has not been set, then the AAAA records (if any) in the parent's delegation should remain as is. If a given address type is set and the child's zone contains no data for that type (as proven by appropriate NSEC or NSEC3 records), then the result in the parent's delegation records for the child should be an empty set.

The procedure for querying for A and AAAA records MUST occur after the procedure, if required, for querying for NS records as defined in Section Section 2.2.2.1. This ensures that the right set NS records is used as provided by the current NS set of the child. I.e, for CSYNC records that have the NS bit set, the NS set used should be the ones pulled from the child during processing. For CSYNC records without the NS bit set, the existing NS records within the parent should be used to determine which A and/or AAAA records to search for.

2.2.2.3. The DNSKEY type

[Editors note: originally I was going to present this draft with no support for DS records as the CDS draft existed and looked like a mechansim that better covered all the DS specific usage cases. However, I've added this section as a discussion point for another mechansim based on conversations with IETF members. CSYNC may or may not be sufficient and opinions are sought about whether CSYNC is a viable mechansim for DS replacement or not.]

The DNSKEY type bit indicates that the DNSKEY records in the child with the SEP bit set and the REVOKE bit cleared should be used to create a new set of DS records for inclusion in the parent's delegation records for the child zone.

A query should be sent to the child zone to obtain all the DNSKEY records within the zone, and DS records should be generated for the appropriate keys.

The DNSKEY type bit MUST NOT be set when the owner/maintainer of the DNSKEY records for a zone that don't contain a set SEP bit is different than the owner/maintainer of the DNSKEY records with the SEP bit set. Organizationally, if the maintainers of the DNSKEY records used to sign the entire contents of the zone are different than the keys intended for the purpose of a secure-entry-point, it is important than only the maintainers of the SEP bit set of DNSKEYs may replace pointers to the SEP bit set of DNSKEYs.

Note: this DS change mechansim does not provide the client with the ability to select (in-band) the DS algorithms used in the parent. The DS type bit should be used instead if both the parent and child wish the child to be able to select the DS algorithm(s) to be used. Children that wish to do so should use the DS type bit instead, if their parental agent supports it.

Note: this DS change mechansim does not let children publish DS records that point to not-yet-published DNSKEYs. Children that wish to do so should use the DS type bit instead, if their parental agent supports it.

2.2.2.4. The DS type

[Editors note: originally I was going to present this draft with no support for DS records as the CDS draft existed and looked like a mechansim that better covered all the DS specific usage cases. However, I've added this section as a discussion point for another mechansim based on conversations with IETF members. CSYNC may or may not be sufficient and opinions are sought about whether CSYNC is a viable mechansim for DS replacement or not.]

The DS type bit indicates that the child has created and published DS records within the child's zone (i.e., below the cut-point) and these records should be copied into the parent's delegation records for the child zone.

A query should be sent to the child zone to obtain all the DS records within the zone, and the DS records should found should be copied into the parent zone's delegation records for the child zone.

The DNSKEY type bit MUST NOT be set when the owner/maintainer of the DNSKEY records for a zone that don't contain a set SEP bit is different than the owner/maintainer of the DNSKEY records with the SEP bit set. Organizationally, if the maintainers of the DNSKEY records used to sign the entire contents of the zone are different than the keys intended for the purpose of a secure-entry-point, it is important than only the maintainers of the SEP bit set of DNSKEYs may replace pointers to the SEP bit set of DNSKEYs.

Parental agents MUST check that at least one of the to-be-published DS records within the new set points to a "secure" DNSSEC-validated DNSKEY record. IE, updating of the DS record set within the parent zone MUST not be done if the parental agent determines it will no longer be possible to validate the zone data within the child. [XXX: yes, a discussion is needed about algorithm and other support differences].

Note: this DS change mechansim does not let the parent select the algorithms for the DS record to be used. The parent MAY choose not to support the DS type bit if they wish to establish policy for DS algorithm usage within their children's zones. In this case, the parental agent should support the DNSKEY type bit instead.

Note: this DS change mechansim lets children publish DS records that may point to unpublished DNSKEY records.

[Editors note: I'm not sure it's safe to reuse the DS record type here. Having two different DS sets at a parent and child is questionably safe from a validator's point of view. It's unclear the effect that it might have on existing deployed code, and it would seem safer to me to publish a new record type, such as the CDS type, for querying child DS records rather than reusing the existing DS type. Opinions desired.]

[One option would be to publish the DS records at a separate location, such as _ds._dns.example.com]

2.3. Operational Considerations

There are a number of important things to consider when deploying a CSYNC RRType.

2.3.1. Error Reporting

There is no inline mechanism for a parental agent to report errors to child zones. Thus, the only error reporting mechanisms must be out of band, such as through a web console or over email. Child operators utilizing the "immediate" flag that fail to see an update

within the parental agent's specified operational window should access the parental agent's log interface to determine why an update failed to be processed.

2.3.2. Child Nameserver Selection

Parental agents will need to poll child nameservers in search of CSYNC records and any other data required for processing a CSYNC record.

Parental agents MAY perform best-possible verification by querying all NS records for available data to determine which has the most recent SOA and CSYNC version (in an ideal world, they would all be equal but this is not possible in practice due to synchronization delays and transfer failures).

Parental agents MAY offer a configuration interface to allow child operators to specify which nameserver should be considered the master to send data queries too. Child operators should be encouraged to make use of this configuration interface.

2.3.3. Documented Parental Agent Type Support

Parental agents that support processing CSYNC records SHOULD publicly document the following minimum processing characteristics:

- The fact that they support CSYNC processing
- The Type Bit Map bits they support
- The frequency with which they poll clients (which MAY be configurable by the client)
- If they support the "immediate" flag
- If they poll a child's single nameserver, a configured list of nameservers, or all of the advertised nameservers when querying records
- If they support SOA serial number caching to avoid issues with regression and/or replay
- Where errors for CSYNC processing are published

2.3.4. Other Considerations

TBD

XXX: Discuss complete replacement scenarios and if allowed.

XXX: Polling frequency discussion

XXX: differences between DS and DNSKEY type bits

3. Security Considerations

TBD

XXX: mention over and over that DNSSEC validation is required for every request

XXX: discussion on DNSSEC checking requirements for both before and after changes take place.

XXX: mention that DS records for SEP-bit DNSKEYs being updating by CSYNC records signed by non-SEP bits should be carefully considered before being used.

4. IANA Considerations

TBD

5. Acknowledgments

A thank you goes out to Warren Kumari and Olafur Gu[eth]mundsson, who's work on the CDS record type helped inspire the work in this document, as well as the definition for "Parental Agent" and "DNS Publisher" definitions. A thank you also goes out to Ed Lewis, who the author held many conversations with about the issues surrounding parent/child relationships and synchronization. Much of the work in this document is derived from the careful existing analysis of these three esteemed colleagues.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, September 2003.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

6.2. Informative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

Author's Address

Wes Hardaker
Parsons, Inc.
P.O. Box 382
Davis, CA 95617
US

Phone: +1 530 792 1913
Email: ietf@hardakers.net