

dnsop  
Internet-Draft  
Intended status: Informational  
Expires: October 13, 2014

W. Kumari  
Google  
O. Gudmundsson  
Shinkuro Inc.  
G. Barwood

April 11, 2014

**Automating DNSSEC Delegation Trust Maintenance**  
**draft-ietf-dnsop-delegation-trust-maintenance-05**

**Abstract**

This document describes a method to allow DNS operators to more easily update DNSSEC Key Signing Keys using the DNS as communication channel. This document does not address the initial configuration of trust anchors for a domain. The technique described is aimed at delegations in which it is currently hard to move information from the child to 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 October 13, 2014.

**Copyright Notice**

Copyright (c) 2014 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 (<http://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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction . . . . .</a>	<a href="#">2</a>
<a href="#">1.1.</a>	<a href="#">Terminology . . . . .</a>	<a href="#">4</a>
<a href="#">1.2.</a>	<a href="#">Requirements Notation . . . . .</a>	<a href="#">4</a>
<a href="#">2.</a>	<a href="#">Background . . . . .</a>	<a href="#">4</a>
<a href="#">2.1.</a>	<a href="#">DNS Delegations . . . . .</a>	<a href="#">4</a>
<a href="#">2.2.</a>	<a href="#">Relationship Between Parent and Child DNS Operator . . .</a>	<a href="#">5</a>
<a href="#">2.2.1.</a>	<a href="#">Solution Space . . . . .</a>	<a href="#">6</a>
<a href="#">2.2.2.</a>	<a href="#">DNSSEC key change process . . . . .</a>	<a href="#">7</a>
<a href="#">3.</a>	<a href="#">CDS / CDNSKEY (Child DS/ Child DNSKEY) Record Definitions . .</a>	<a href="#">7</a>
<a href="#">3.1.</a>	<a href="#">CDS Resource Record Format . . . . .</a>	<a href="#">8</a>
<a href="#">3.2.</a>	<a href="#">CDNSKEY Resource Record Format . . . . .</a>	<a href="#">8</a>
<a href="#">4.</a>	<a href="#">Automating DS Maintenance With CDS/CDNSKEY records . . . . .</a>	<a href="#">8</a>
<a href="#">4.1.</a>	<a href="#">CDS / CDNSKEY Processing Rules . . . . .</a>	<a href="#">9</a>
<a href="#">5.</a>	<a href="#">CDS / CDNSKEY Publication . . . . .</a>	<a href="#">9</a>
<a href="#">6.</a>	<a href="#">Parent Side CDS / CDNSKEY Consumption . . . . .</a>	<a href="#">9</a>
<a href="#">6.1.</a>	<a href="#">Detecting a Changed CDS / CDNSKEY . . . . .</a>	<a href="#">10</a>
<a href="#">6.1.1.</a>	<a href="#">CDS / CDNSKEY Polling . . . . .</a>	<a href="#">10</a>
<a href="#">6.1.2.</a>	<a href="#">Other Mechanisms . . . . .</a>	<a href="#">10</a>
<a href="#">6.2.</a>	<a href="#">Using the New CDS / CDNSKEY Records . . . . .</a>	<a href="#">11</a>
<a href="#">6.2.1.</a>	<a href="#">Parent Calculates DS . . . . .</a>	<a href="#">11</a>
<a href="#">7.</a>	<a href="#">IANA Considerations . . . . .</a>	<a href="#">12</a>
<a href="#">8.</a>	<a href="#">Privacy Considerations . . . . .</a>	<a href="#">12</a>
<a href="#">9.</a>	<a href="#">Security Considerations . . . . .</a>	<a href="#">12</a>
<a href="#">10.</a>	<a href="#">Acknowledgements . . . . .</a>	<a href="#">14</a>
<a href="#">11.</a>	<a href="#">References . . . . .</a>	<a href="#">14</a>
<a href="#">11.1.</a>	<a href="#">Normative References . . . . .</a>	<a href="#">14</a>
<a href="#">11.2.</a>	<a href="#">Informative References . . . . .</a>	<a href="#">15</a>
<a href="#">Appendix A.</a>	<a href="#">RRR background . . . . .</a>	<a href="#">15</a>
<a href="#">Appendix B.</a>	<a href="#">Changes / Author Notes. . . . .</a>	<a href="#">16</a>
	<a href="#">Authors' Addresses . . . . .</a>	<a href="#">18</a>

## [1.](#) Introduction

When a DNS operator first signs their zone, they need to communicate their DS record(s) (or DNSKEY(s)) to their parent through some out-of-band method to complete the chain of trust.

Each time the child changes/rolls the key that is represented in the parent, the new and/or deleted key information has to be communicated to the parent and in the parent's zone. How this information is sent to the parent depends on the relationship the child has with the



parent. In many cases this is a manual process, and not an easy one. For each key roll, there may be up to two interactions with the parent. Any manual process is susceptible to mistakes and/or errors. In addition, due to the annoyance factor of the process, operators may avoid performing key rollovers or skip needed steps to publish the new DS at the parent.

DNSSEC provides data integrity to information published in DNS; thus DNS publication can be used to automate maintenance of delegation information. This document describes a method to automate publication of subsequent DS records, after the initial one has been published.

Readers are expected to be familiar with DNSSEC, including [[RFC4033](#)], [[RFC4034](#)], [[RFC4035](#)], [[RFC5011](#)] and [[RFC6781](#)].

This document is a compilation of two earlier drafts: [draft-barwood-dnsop-ds-publish](#)[[I-D.ds-publish](#)] and [draft-wkumari-dnsop-ezkeyroll](#).

This document outlines a technique in which the parent periodically (or upon request) polls its signed children and automatically publish new DS records. To a large extent, the procedures this document follows are as described in [[RFC6781](#)] [section 4.1.2](#).

This technique is designed to be friendly both to fully automated tools and humans. Fully automated tools can perform all the actions needed without human intervention, and thus can monitor when it is safe to move to the next step.

The solution described in this document only allows transferring information about DNSSEC keys (DS and DNSKEY) from the child to the parental agent. It lists exactly what the parent should publish, and allows for publication of stand-by keys. A different protocol, [[I-D.csync](#)], can be used to maintain other important delegation information, such as NS and glue. These two protocols have been kept as separate solutions because the problems are fundamentally different, and a combined solution is overly complex.

This document describes a method for automating maintenance of the delegation trust information, and proposes a polled / periodic trigger for simplicity. Some users may prefer a different trigger, such as a button on a webpage, a REST interface, DNS NOTIFY, etc. These alternate / additional triggers are not discussed in this document.

This proposal does not include all operations needed for the maintenance of DNSSEC key material, specifically the initial introduction or complete removal of all keys. Because of this,



alternate communications mechanisms must always exist, potentially introducing more complexity.

### **1.1. Terminology**

The terminology we use is defined in this section.

Highlighted roles:

- o Child: "The entity on record that has the delegation of the domain from the parent"
- o Parent: "The domain in which the child is registered"
- o Child DNS Operator: "The entity that maintains and publishes the zone information for the child DNS"
- o Parent DNS Operator: "The entity that maintains and publishes the zone information for the parent DNS"
- o Parental Agent: "The entity that the child has relationship with, to change its delegation information"
- o Provisioning system: "A system that the operator of the master DNS server operates to maintain the information published in the DNS. This includes the systems that sign the DNS data"

RRR is our shorthand for Registry/Registrar/Registrant model of parent child relationship see [Appendix A](#) for more.

### **1.2. Requirements Notation**

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](#)].

## **2. Background**

### **2.1. DNS Delegations**

DNS operation consists of delegations of authority. For each delegation there are (most of the time) two parties: the parent and the child.

The parent publishes information about the delegations to the child; for the name servers it publishes an NS [[RFC1035](#)] RRset that lists a hint for name servers that are authoritative for the child. The



child also publishes a NS RRset, and this set is the authoritative list of name servers to the child zone.

The second RRset the parent sometimes publishes is the DS [[RFC4034](#)] set. The DS RRset provides information about the DNSKEY(s) that the child has told the parent it will use to sign its DNSKEY RRset. In DNSSEC trust relationship between zones is provided by the following chain:

parent DNSKEY --> DS --> child DNSKEY.

A prior proposal [[I-D.auto-cpsync](#)] suggested that the child send an "update" to the parent via a mechanism similar to Dynamic Update. The main issue became: How does the child find the actual parental agent/server to send the update to? While that could have been solved via technical means, the proposal died. There is also a similar proposal in [[I-D.parent-zones](#)].

As the DS record can only be present at the parent ([RFC4034](#) [[RFC4034](#)]), some other record/method is needed to automate which DNSKEYs are picked to be represented in the parent zone's DS records. One possibility is to use flags in the DNSKEY record. If the SEP bit is set, this indicates that the DNSKEY is intended for use as a secure entry point. This DNSKEY signs the DNSKEY RRset, and the Parental Agent can calculate DS records based on that. But this fails to meet some operating needs, including the child having no influence what DS digest algorithms are used and DS records can only be published for keys that are in the DNSKEY RRset, and thus this technique would not be compatible with Double-DS key rollover.

## **[2.2.](#) Relationship Between Parent and Child DNS Operator**

In practical application, there are many different relationships between the parent and child DNS operators. The type of relationship affects how the Child DNS Operator communicates with the parent. This section will highlight some of the different situations, but is by no means a complete list.

Different communication paths:

- o Direct/API: The child can change the delegation information via automated/scripted means EPP[[RFC5730](#)] used by many TLDs is an example of this. Another example is the web service's programmatic interfaces that Registrars make available to their Resellers.
- o User Interface: The Child uses a (web) site set up by the Parental Agent for updating delegation information.





- o Indirect: The communication has to be transmitted via out-of-band between two parties, such as email, telephone etc.. This is common when the Child's DNS operator is neither the child itself nor the Registrar for the domain but a third party.
- o Multi-step Combinations: The information flows through an intermediary. It is possible, but unlikely, that all the steps are automated via API's and there are no humans involved.

A domain name holder (Child) may operate its own DNS servers or outsource the operation. While we use the word parent as a singular, parent can consist of single entity or a composite of many discrete parts that have rules and roles. We refer to the entity that the child corresponds with as the Parent.

In another common case an enterprise may delegate parts of its name-space to be operated by a group that is not the same as that which operates the enterprise's DNS servers. In this case the flow of information is frequently handled in either an ad hoc manner or via some corporate mechanism; this can range from email to fully-automated operation. The word enterprise above covers all organizations in which the domains are not sold on the open market and there is some relationship between the entities.

#### **2.2.1. Solution Space**

This document is aimed at the cases in which there is an organizational separation of the child and parent.

A further complication is when the Child DNS Operator is not the Child. There are two common cases of this:

- a) The Parental Agent (e.g. registrar) handles the DNS operation.
- b) A third party takes care of the DNS operation.

If the Parental Agent is the DNS operator, life is much easier; the Parental Agent can inject any delegation changes directly into the Parent's Provisioning system. The techniques described below are not needed in the case when Parental Agent is the DNS operator.

In the case of a third party DNS operator, the Child either needs to relay changes in DNS delegation or give the Child DNS Operator access to its delegation/registration account.

Some parents want the child to express their DNSKEYS in the form of DS records, while others want to receive the DNSKEY records and calculate the DS records themselves. There is no consensus on which



method is better; both have good reasons to exist. The proposal below can operate with both models, but the child needs to be aware of the parental policies.

#### **2.2.2. DNSSEC key change process**

After a Child DNS Operator first signs the zone, there is a need to interact with the Parent, for example via the delegation account interface, to "upload / paste-in the zone's DS information". The action of logging in through the delegation account user interface authenticates that the user is authorized to change delegation information for the child published in the parent zone. In the case where the "Child DNS Operator" does not have access to the registration account, the Child needs to perform the action.

At a later date, the Child DNS Operator may want to publish a new DS record in the parent, either because they are rolling keys, or because they want to publish a stand-by key. This involves performing the same process as before. Furthermore when this is a manual process with cut and paste, operational mistakes will happen -- or worse, the update action is not performed at all.

The Child DNS Operator may also introduce new keys, and can do so when old keys exist and can be used. The Child may also remove old keys, but this document does not support removing all keys. This is to avoid making signed zones unsigned. The Child may not enroll the initial key or introduce a new key when there are no old keys that can be used, because there is no way to validate the information.

### **3. CDS / CDNSKEY (Child DS/ Child DNSKEY) Record Definitions**

This document specifies two new resource records, CDS and CDNSKEY. These records are used to convey, from one zone to its parent, the desired contents of the zone's DS resource record set residing in the parent zone.

The CDS / CDNSKEY records are published in the child zone and gives the child control of what is published for it in the parental zone. The CDS / CDNSKEY RRset expresses what the child would like the DS RRset to look like after the change; it is a "replace" operation, and it is up to the consumer of the records to translate that into the appropriate add/delete operations in the registration systems (and in the case of CDNSKEY, to generate the DS from the DNSKEY). If no CDS / CDNSKEY RRset is present in child, this means that no change is needed.

[[RFC Editor: Please remove this paragraph before publication]  
Version -04 of the ID that became this working group document (<http://>



/tools.ietf.org/id/draft-kumari-ogud-dnsop-cds-04.txt) defined a new record (CTA) that could hold either a DS or a DNSKEY record (with a selector to differentiate between them). In a shocking development, there was almost full consensus that this was horrid :- ) ]

### **3.1. CDS Resource Record Format**

The wire and presentation format of the CDS ("Child DS") record is identical to the DS record [[RFC4034](#)]. IANA has allocated RR code 59 for the CDS record via expert review [[I-D.ds-publish](#)]. CDS uses the same registries as DS for its fields.

No special processing is performed by authoritative servers or by revolvers, when serving or resolving. For all practical purposes CDS is a regular RR type.

### **3.2. CDNSKEY Resource Record Format**

The wire and presentation format of the CDNSKEY ("Child DNSKEY") record is identical to the DNSKEY record. CDNSKEY uses the same registries as DNSKEY for its fields.

No special processing is performed by authoritative servers or by revolvers, when serving or resolving. For all practical purposes CDNSKEY is a regular RR type.

## **4. Automating DS Maintenance With CDS/CDNSKEY records**

CDS/CDNSKEY records are intended to be "consumed" by delegation trust maintainers. The use of CDS/CDNSKEY is optional.

Some parents prefer to accept DNSSEC key information in DS format, some parents prefer to accept it in DNSKEY format, and calculate the DS record on the child's behalf. Each method has pros and cons, both technical and policy. This solution is DS vs DNSKEY agnostic, and allows operation with either.

The child SHOULD publish both CDS and CDNSKEY records. If the child knows which the parent consumes, it MAY choose to only publish that record type (for example, some children wish the parent to publish a DS, but they wish to keep the DNSKEY "hidden" until needed). If the child publishes both, the two RRsets MUST match in content. The parent should use whichever one they choose, but MUST NOT query for both and perform consistency checks between the CDS and CDNSKEY records.



#### **[4.1.](#) CDS / CDNSKEY Processing Rules**

If there are no CDS / CDNSKEY resource records in the child, this signals that no change should be made to the current DS set. This means that, once the child and parent are in sync, the child DNS operator SHOULD remove all CDS records from the zone.

Following acceptance rules are placed on the CDS / CDNSKEY records as follows:

- o Location: "the CDS / CDNSKEY record MUST be at the child zone apex".
- o Signer: "MUST be signed with a key that is represented in both the current DNSKEY and DS RRset's."
- o Continuity: "MUST NOT break the current delegation if applied to DS RRset"

If any these conditions fail the CDS / CDNSKEY record MUST be ignored.

#### **[5.](#) CDS / CDNSKEY Publication**

Child DNS Operator SHOULD only publish a CDS or CDNSKEY RRset when it wants to make a change to the DS RRset in the Parent. In order to be valid, the CDS / CDNSKEY RRset MUST be compliant with the rules in [Section 4.1](#). When the Parent DS is "in-sync" with the CDS / CDNSKEY, the Child DNS Operator SHOULD delete the CDS / CDNSKEY RRset(s). Note that if the child has published a CDNSKEY RR, the Parent will have to calculate the DS (using the requested digest algorithm) to do the comparison.

#### **[6.](#) Parent Side CDS / CDNSKEY Consumption**

The CDS / CDNSKEY RRset SHOULD be used by the Parental Agent to update the DS RRset in the parent zone. The Parental Agent for this uses a tool that understands the CDS / CDNSKEY signing rules from [Section 4.1](#) so it may not be able to use a standard validator.

The parent MUST choose to accept either CDS or CDNSKEY records (based upon local policy), and MUST NOT expect there to be both. A parent MUST NOT perform a consistency check between CDS and CDNSKEY (other than for informational / debugging use).





### **6.1. Detecting a Changed CDS / CDNSKEY**

How the Parental Agent gets the CDS / CDNSKEY record may differ, below are two examples as how this can take place.

**Polling** The Parental Agent operates a tool that periodically checks each of the children that has a DS record to see if there is a CDS or CDNSKEY record.

**Pushing** The delegation user interface has a button {Fetch DS} when pushed preforms the CDS / CDNSKEY processing. If the Parent zone does not contain DS for this delegation then the "push" SHOULD be ignored. If the Parental Agent displays the contents of the CDS / CDNSKEY to the user and gets confirmation that this represents their key, the Parental Agent MAY use this for initial enrolment (when the Parent zone does not contain the DS for this delegation).

In either case the Parental Agent MAY apply additional rules that defer the acceptance of a CDS / CDNSKEY change, these rules may include a condition that the CDS / CDNSKEY remains in place and valid for some time period before it is accepted. It may be appropriate in the "Pushing" case to assume that the Child is ready and thus accept changes without delay.

#### **6.1.1. CDS / CDNSKEY Polling**

This is the only defined use of CDS / CDNSKEY in this document. There are limits to the saleability of polling techniques, thus some other mechanism is likely to be specified later that addresses CDS / CDNSKEY usage in the situation where polling does not scale to. Having said that Polling will work in many important cases like enterprises, universities, small TLDs etc. In many regulatory environments the registry is prohibited from talking to the registrant. In most of these cases the registrant has a business relationship with the registrar, and so the registrar can offer this as a service.

If the CDS / CDNSKEY RRset does not exist, the Parental Agent MUST take no action. Specifically it MUST NOT delete or alter the existing DS RRset.

#### **6.1.2. Other Mechanisms**

It is assumed that other mechanisms will be implemented to trigger the parent to look for an updated CDS / CDNSKEY record. As the CDS / CDNSKEY records are validated with DNSSEC, these mechanisms can be unauthenticated (for example, a child could telephone its parent and



request that they process the new CDS or CDNSKEY record, an unauthenticated POST could be made to a webserver (with rate-limiting), etc.)

Other documents can specify the trigger conditions.

## **6.2. Using the New CDS / CDNSKEY Records**

Regardless of how the Parental Agent detected changes to a CDS / CDNSKEY RR, the Parental Agent MUST use a DNSSEC validator to obtain a validated CDS / CDNSKEY RRset from the Child zone.

The Parental Agent MUST ensure that old versions of the CDS / CDNSKEY RRset do not overwrite newer versions. This MAY be accomplished by checking that the signature inception in the RRSIG for CDS / CDNSKEY is newer and/or the serial number on the child's SOA is greater. This may require the Parental Agent to maintain some state information.

The Parental Agent MAY take extra security measures. For example, to mitigate the possibility that a Child's key signing key has been compromised, the Parental Agent may, for example, inform (by email or other methods ) the Child DNS Operator of the change. However the precise out-of-band measures that a parent zone SHOULD take are outside the scope of this document.

Once the Parental Agent has obtained a valid CDS / CDNSKEY it MUST check the publication rules from [section 4.1](#). In particular the Parental Agent MUST double check the Continuity rule and do its best not to invalidate the Child zone. Once checked and if the CDS / CDNSKEY and DS differ it may apply the changes to the parent zone. If the parent consumes CDNSKEY, the parent should calculate the DS before doing this comparison.

### **6.2.1. Parent Calculates DS**

There are cases where the Parent wants to calculate the DS record due to policy reasons. In this case, the Child publishes CDNSKEY records and the parent calculates the DS records on behalf of the children.

When a Parent operates in "calculate DS" mode it can operate in one of two sub-modes

full i.e. it only publishes DS records it calculates from DNSKEY records,

augment i.e. it will make sure there are DS records for the digest algorithm(s) it requires(s).



In the case the parent fetch the CDNSKEY and calculate the DS it MAY be the case that the DS published in the parent zone is not identical with the data in the CDS record made available by the child.

## **7. IANA Considerations**

IANA has assigned RR Type code 59 for CDS. This was done for an earlier version of this document[I-D.ds-publish] This document is to become the reference for CDS RRtype.

IANA is requested to assign another RR Type for the CDNSKEY.

## **8. Privacy Considerations**

All of the information handled / transmitted by this protocol is public information published in the DNS.

## **9. Security Considerations**

This work is for the normal case; when things go wrong there is only so much that automation can fix.

If child breaks DNSSEC validation by removing all the DNSKEYs that are represented in the DS set its only repair actions are to contact the parent or restore the DNSKEYs in the DS set.

In the event of a compromise of the server or system generating signatures for a zone, an attacker might be able to generate and publish new CDS records. The modified CDS records will be picked up by this technique and so may allow the attacker to extend the effective time of his attack. If there a delay in accepting changes to DS, as in [[RFC5011](#)], then the attacker needs to hope his activity is not detected before the DS in parent is changed. If this type of change takes place, the child needs to contact the parent (possibly via a registrar web interface) and remove any compromised DS keys.

A compromise of the account with the parent (e.g. registrar) will not be mitigated by this technique, as the "new registrant" can delete/modify the DS records at will.

While it may be tempting, this SHOULD NOT be used for initial enrollment of keys since there is no way to ensure that the initial key is the correct one. If is used, strict rules for inclusion of keys like hold down times, challenge data inclusion etc., ought to be used, along with some kind of challenge mechanism. A child cannot use this mechanism to go from signed to unsigned (publishing an empty CDS / CDNSKEY RRset means no-change should be made in the parent).



The CDS RR type should allow for enhanced security by simplifying process. Since rollover is automated, updating a DS RRset by other means may be regarded as unusual and subject to extra security checks.

As this introduces a new mechanism to update information in the parent it MUST be clear who is fetching the records and creating the appropriate records in the parent zone. Specifically some operations may use other mechanisms than what is described here. For example, a registrar may assume that it is maintaining the DNSSEC key information in the registry, and may have this cached. If the registry is fetching the CDS / CDNSKEY then the registry and registrar may have different views of the DNSSEC key material and the result of such a situation is unclear. Because of this, this mechanism SHOULD NOT be used to bypass intermediaries that might cache information and because of that get the wrong state.

If there is a failure in applying changes in child zone to all DNS servers listed in either parent or child NS set it is possible that the Parental agent may get confused either not perform action because it gets different answers on different checks or CDS validation fails. In the worst case, Parental Agent performs an action reversing a prior action but after the child signing system decides to take the next step in rollover, resulting in a broken delegation.

DNS is a loosely coherent distributed database with local caching; therefore, it is important to allow old information to expire from caches before deleting DS or DNSKEY records. Similarly, it is important to allow new records to propagate through the DNS before use, see [[RFC6781](#)] and [[I-D.key-time](#)]

It is common practice for users to outsource their DNS hosting to a 3rd party DNS provider. In order for that provider to be able to maintain the DNSSEC information some users give the provider their registrar login credentials (which obviously has negative security implications). Deploying the solution described in this document allows the 3rd party DNS provider to maintain the DNSSEC information without giving them the registrar credentials, thereby improving security.

By automating the maintenance of the DNSSEC key information (and removing humans from the process) we expect to decrease the number of DNSSEC related outages, which should increase DNSSEC deployment.





## **10. Acknowledgements**

We would like to thank a large number of folk, including: Mark Andrews, Joe Abley, Jaap Akkerhuis, Roy Arends, Doug Barton, Brian Dickinson, Paul Ebersman, Tony Finch, Jim Galvin, Paul Hoffman, Samir Hussain, Tatuya Jinmei, Olaf Kolkman, Cricket Liu, Stephan Lagerholm, Matt Larson, Marco Sanz, Antoin Verschuren, Suzanne Woolf, Paul Wouters, John Dickinson, Timothe Litt and Edward Lewis.

Special thanks to Wes Hardaker for contributing significant text and creating the complementary (CSYNC) solution, and to Patrik Faltstrom, Paul Hoffman, Matthijs Mekking and Mukund Sivaraman for text and in-depth review.

There were a number of other folk with whom we discussed this, apologies for not remembering everyone.

## **11. References**

### **11.1. Normative References**

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, [RFC 1035](#), November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", [RFC 4033](#), March 2005.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", [RFC 4034](#), 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.
- [RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, [RFC 5011](#), September 2007.
- [RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", [RFC 6781](#), December 2012.



## **11.2. Informative References**

- [I-D.auto-cpsync]  
Mekking, W., "Automated (DNSSEC) Child Parent Synchronization using DNS UPDATE", [draft-mekking-dnsop-auto-cpsync-01](#) (work in progress), December 2010.
- [I-D.csync]  
Hardaker, W., "Child To Parent Synchronization in DNS", [draft-hardaker-dnsop-csync-02](#) (work in progress), July 2013.
- [I-D.ds-publish]  
Barwood, G., "DNS Transport", [draft-barwood-dnsop-ds-publish-02](#) (work in progress), June 2011.
- [I-D.key-time]  
Mekking, W., "DNSSEC Key Timing Considerations", [draft-ietf-dnsop-dnssec-key-timing-03](#) (work in progress), July 2012.
- [I-D.parent-zones]  
Andrews, M., "Updating Parent Zones", November 2013.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, [RFC 5730](#), August 2009.
- [RFC5910] Gould, J. and S. Hollenbeck, "Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)", [RFC 5910](#), May 2010.

## **Appendix A. RRR background**

In the RRR world, the different parties are frequently from different organizations. In the single enterprise world there are also organizational/geographical/cultural separations that affect how information flows from a Child to the parent.

Due to the complexity of the different roles and interconnections, automation of delegation information has been punted in the past. There have been some proposals to automate this, in order to improve the reliability of the DNS. These proposals have not gained enough traction to become standards.

For example in many of the TLD cases there is the RRR model (Registry, Registrar and Registrant). The Registry operates DNS for the TLD, the Registrars accept registrations and place information into the Registries database. The Registrant only communicates with



the Registrar; frequently the Registry is not allowed to communicate with the Registrant. In that case as far as the registrant is concerned the Registrar == Parent.

In many RRR cases the Registrar and Registry communicate via EPP[RFC5730] and use the EPP DNSSEC extension [[RFC5910](#)]. In a number of ccTLDs there are other mechanisms in use as well as EPP, but in general there seems to be a movement towards EPP usage when DNSSEC is enabled in the TLD.

## [Appendix B](#). Changes / Author Notes.

[RFC Editor: Please remove this section before publication ]

WG-04 to WG-05

- o More comments from Patrik, Paul and Ed.
- o Many nits and fixes from Matthijs Mekking.
- o Outstanding question: Should we remove the "Child SHOULD remove the CDS record" text? Mail sent to list.

WG-03 to WG-04

- o Large number of comments and changes from Patrik.

WG-02 to WG-03

- o Fixed some references to [RFC 5011](#) - from Samir Hussain.
- o Fixed some spelling / typos - from Samir Hussain.
- o A number of clarifications on the meaning on an empty / non-existent CDS RRset - thanks to JINMEI, Tatuya
- o Be consistent in mentioning both CDS and CDNSKEY throughout the document.

WG-01 to WG-02

- o Many nits and suggestions from Mukund.
- o Matthijs: " I still think that it is too strong that the Child DNS Operator SHOULD/MUST delete the CDS RRset when the Parent DS is "in-sync". This should be a MAY"

WG-00 to WG-01



- o Addressed Vancouver: "Paul Hoffmann: NOT ready for WGLC. None of the 2 documents explain why there is a split between the two strategies." Thanks to Paul for providing text.

From -05 to WG-00:

- o Nothing rchanged, resubmit under new name.

From 04 to 05

- o Renamed the record back to CDS.

From 03 to 04.

- o Added text explaining that CDS and CSYNC complement each other, not replace or compete.
- o Changed format of record to be <selector> <data> to allow the publication of DS \*\*or\*\* DNSKEY.
- o Bunch of text changed to cover the above.
- o Added a bit more text on the polling scaling stuff, expectation that other triggers will be documented.

From 02 to 03

- o Applied comments by Matthijs Mekking
- o Incorporated suggestions from Edward Lewis about structure
- o Reworked structure to be easier for implementors to follow
- o Applied many suggestions from a wonderful thorough review by John Dickinson
- o Removed the going Unsigned option

From 01 to 02

- o Major restructuring to facilitate easier discussion
- o Lots of comments from DNSOP mailing list incorporated, including making draft DNSKEY/DS neutral, explain different relationships that exists,
- o added more people to acks.





- o added description of enterprise situations
- o Unified on using Parental Agent over Parental Representative
- o Removed redundant text when possible
- o Added text to explain what can go wrong if not all child DNS servers are in sync.
- o Reference prior work by Matthijs Mekking
- o Added text when parent calculates DS from DNSKEY

From - to -1.

- o Removed from section .1: "If a child zone has gone unsigned, i.e. no DNSKEY and no RRSIG in the zone, the parental representative MAY treat that as intent to go unsigned. (NEEDS DISCUSSION)."  
Added new text at end. -- suggestion by Scott Rose 20/Feb/13.
- o Added some background on the different DNS Delegation operating situations and how they affect interaction of parties. This moved some blocks of text from later sections into here.
- o Number of textual improvements from Stephan Lagerholm
- o Added motivation why CDS is needed in CDS definition section
- o Unified terminology in the document.
- o Much more background

#### Authors' Addresses

Warren Kumari  
Google  
1600 Amphitheatre Parkway  
Mountain View, CA 94043  
US

Email: warren@kumari.net



Olafur Gudmundsson  
Shinkuro Inc.  
4922 Fairmont Av, Suite 250  
Bethesda, MD 20814  
USA

Email: [ogud@ogud.com](mailto:ogud@ogud.com)

George Barwood  
33 Sandpiper Close  
Gloucester GL2 4LZ  
United Kingdom

Email: [george.barwood@blueyonder.co.uk](mailto:george.barwood@blueyonder.co.uk)

