# Child ---> Parent DNS Data Synchronization

*A Panel of Options*
*By a Panel of People*

Wes Hardaker
Olafur Gudmundsson
Warren Kumari

# What Is The Problem?

- Parent has "linking" records: *(possibly more in the future??)*
  - DS (authoritative)
  - NS (hint)
  - Glue: A / AAAA (hint)
    - Kinds: In-child, In-sibling, outside parent
- State of the world today:
  - Very few children can use EPP directly
  - Manual action to push changes to the parent
    - Everyone gets it wrong sometime
      - Too lazy/painful
      - Just forget/make mistakes
      - Parent policies force "lies"

# What are The Consequences?

- When the DS is incorrect
  - The zone can go bogus
- NS and Glue errors
  - Correct servers:
    - If not listed: They *don't get any traffic*
    - If NSset mismatch: *Traffic skewed*
  - Incorrect addresses: Get traffic they shouldn't
  - DNS resolution is slower and/or brittle

# What Can We Do?

- Define standard mechanisms to:
  - "Martial the data" -- Ed Lewis
  - Perform operations (i.e. act on the martialed data)
- Proposed Solutions to date:
  - CDS: *Specifies DS content in the child*
  - CSYNC: Bit-*pointers that specify what to copy*
- Other considerations:
  - Out of band options

    HTTP/REST EPP

    Netconf etc…

# Design Requirements

- Child
  - Needs to publish data upward
  - Should be able to say “act now”
- Parental Agent
  - Needs to securely obtain the data
  - Needs to be able to control their policies
    - MAY only accept data of certain types (DNSKEY vs DS)
    - MAY require a manual operator “push this button”
      - (likely an out of band button)
    - MAY or MAY NOT keep state

# Design Choices: What Protocol To Use

- Both CDS and CSYNC use DNS:
  - A pull model (by default)
  - Don’t need to build another security model
  - No changes needed to DNS servers and resolvers
    - Except for the new types
    - Changes are limited to provisioning systems
  - Parental agent schedules pulls and applies policies.
- There are other choices, of course
  - Some are push models

# Design Choices: How To Publish

- Publish data in separate records
  - CDS: Child publishes a “copy me” set
- Publish a pointer to the data
  - CSYNC: indicates what records to copy

# Design Choices: Where To Publish

- Using new records:
  - At the zone cut?
  - At special places
    - _dnssec.example.com
      - Does not work with DNAME @ apex
- Using existing records:
  - copy from existing child NS/A/AAAA

# Approach: CDS

- CDS publishes the DS records to copy
- CDS is simply a new record type

- CDS requires a signature by:
  - Key(s) in current DS & DNSKEY set

# Tricky Issues: DS Records

- DS records point to only some keys
  - They're signed by the parent
  - The child could sign a new set, but with what?
  - Should the other keys be allowed to update it?
- DS records are derived from keys
  - Some parents only want keys
  - Some children need to pre-publish

# Approach: CSYNC

- CSYNC publishes:
  - A list of things to copy
    - *NS/A/AAAA/DS/DNSKEY*
  - A minimum SOA serial number to copy from
  - A policy bit indicating whether the child means:
    - Act now
    - Copy in and wait for a button push
- Is the above KISS?
  - If not, what should go?
- CSYNC discusses:
  - Synchronization issues if SOA changes during pull

# WG Directional Questions

- What to do with:
  - The CDS draft
  - The CSYNC draft
- Keep the method always the same?
  - Or is DS different and important to be separate
- Use both?
  - Should CSYNC be used to signal CDS existence?
  - Or is the CDS existence itself be the bit?