Critical Zone Managers’ Implementation Guidelines

DNS Security

1. **Practice 1:** Authoritative zones **MUST** be DNSSEC signed and best practices for key management **MUST** be followed.

   If you are a ccTLD zone manager/operator, the ICANN DNSSEC guidebook for ccTLDs may be helpful.

   General Implementation Considerations:

   Actually getting a zone DNSSEC signed requires the following steps. Note: Two pairs of public/private keys are typically used: a Zone Signing Key (ZSK), which is used to sign zone records themselves (zone data), and a Key Signing Key (KSK), which only signs the ZSK.

   1. Generate keys - usually a ZSK and KSK as described above
   2. Include the public part of the ZSK and KSK in the zone (DNSKEY RRset)
   3. Decide on a scheme for proof of non-existence: NSEC or NSEC3
   4. Sign the zone data using the private part of the ZSK
      - For all records in the zone, including TTL value and type, a digital signature is created (hash data + encrypt with ZSK private key)
      - RRSIG records are added for each RRset (same name + type)
      - NSEC/NSEC3 records are added
   5. Sign the DNSKEY RRset with the KSK

   Once the above has been done, the KSK must be communicated to the managers of the parent zone so that it (actually the hash, in the form of a DS record) can be added to the parent zone and signed using the parent zone’s ZSK.

   Signatures have a fixed lifetime and must be renewed/recreated before they expire. Typically, the validity of signatures is anywhere between two and four weeks.

   Note that most modern DNS authoritative software is able to do all this for you. In the notes below, we cover the simplest possible configuration for a fully automated DNSSEC zone signing. Also, be aware that it’s not mandatory to have two keypairs (ZSK and KSK); it’s also possible to use a Single Type Signing Scheme, with one Combined Signing Key (CSK) used for signing records and also deriving the DS record to be included in the parent zone. While there are advantages to using a single key (CSK), it does mean that rolling the key for signing zone records forces an update of the DS in the parent zone as well.

Note that if managing a TLD, management of the DS, just like the NS, is done using the IANA RZM portal.

BIND:

The current preferred method for signing zones with BIND is described at https://bind9.readthedocs.io/en/v9_18_5/dnssec-guide.html, as well as https://kb.isc.org/docs/dnssec-key-and-signing-policy.

Here is an example for a fully automated DNSSEC zone signing setup with BIND9. It assumes that BIND has write access to the zone directory so it can make the necessary updates to the zone journal and manage keys.

```
zone "example.com" {
      type primary;
      file "...";
      dnssec-policy default;
};
```

From the BIND documentation:

> *The `dnssec-policy` statement causes the zone to be signed and turns on automatic maintenance for the zone. This includes re-signing the zone as signatures expire and replacing keys on a periodic basis. The value `default` selects the default policy, which contains values suitable for most situations.*

Note: this is a very minimal setup. Please consult the full BIND9 DNSSEC Guide (above link). Also note that in this case, a single key is used (CSK).

PowerDNS:

PowerDNS has several modes of operations. Online (or live) and front-signing are most relevant. The simplest is by far "Online mode," where PowerDNS manages all zone data and key material, and signatures are generated on-the-fly. This is the default mode, and makes the most sense if you are already managing your zone data using PowerDNS.

Front-signing enables PowerDNS to act as a "front-end" authoritative server, fetching an unsigned zone from a primary server. PowerDNS handles all key material and signs the records on-the-fly, as for online mode. This is especially useful if the zone is being served by a legacy DNS platform that isn't DNSSEC-capable and cannot easily be migrated or replaced (e.g., older Windows DNS).

PowerDNS can also be configured to act as a “hidden SOA,” where the zone is signed but PowerDNS isn’t serving the zone to the wider Internet. Instead, one or more authoritative servers are configured as secondaries to the zone, fetching the primary copy from the PowerDNS Online signing instance.

For more details, see the PowerDNS Authoritative Nameserver reference.

Windows DNS:

Unfortunately, Microsoft no longer offers up-to-date guidance on deploying DNSSEC on Windows Server. The last update is for Windows 2012r2, published in 2016. As a result, we cannot currently provide BCPs on deploying DNSSEC for authoritative zones on Windows Server.

2. **Practice 2:** Access to zone transfer between authoritative servers **MUST** be limited. Configure ACLs and TSIG in the DNS Authoritative software package to restrict zone transfers to secondary servers only.

   General Implementation Considerations:

   Limit access to AXFR/IXFR so that only secondary DNS servers are allowed to request a copy of the zone, as well as any other authorized third parties. This may be for research purposes, monitoring/verification of the publication of zone content, etc.

   BIND:

   At a minimum, use IP access lists to restrict who may transfer zones:

   ```
   acl “secondaries” {
     1.2.3.4;
     10.20.30.0/24;
     localnets;
   };

   options {
     allow-transfer { secondaries; 127.0.0.1; ::1; };
   };
   ```

   It’s recommended to use TSIG, and to use a key for each pair of host:

   ```
   # tsig-keygen them-us
   ```

   Output:

```
key "them-us" {
  algorithm hmac-sha256;
  secret "xw5I/zKwmJHQulWA3vDNGn4kdBNgsmIhqrjxqMvv5Cs=";
};

zone “example.com” {
  type master;
  file “.../example.com”;
  allow-transfer { key “them-us”; };
};
```

Note that keys can also be used in `allow-transfer` in the `options` section. (applies to all zones unless specified individually).

PowerDNS:

In PowerDNS Authoritative, per-zone ACLs are managed using the `pdnsutil set-meta` command. An example would be:

```
$ pdnsutil set-meta example.org ALLOW-AXFR-FROM AUTO-NS
192.0.2.0/24
```

In this example, the subnet 192.0.2.0/24 as well as all the listed secondary servers for the zone (defined by NS records in the zone) would be authorized to perform a zone transfer of “example.org”. `AUTO-NS` is a special value meaning “all secondary authoritative servers”.

More info is available at the [PowerDNS Authoritative documentation](#) site.

To implement TSIG for outbound AXFR access (i.e.: PowerDNS Authoritative will only allow secondaries that use a matching TSIG key to perform a zone transfer), do the following:

```
$ pdnsutil import-tsig-key them-us hmac-md5
     'xw5I/zKwmJHQulWA3vDNGn4kdBNgsmIhqrjxqMvv5Cs='
$ pdnsutil activate-tsig-key example.org them-us master
```

As noted in the PowerDNS Authoritative documentation for [configuring TSIG](#), any host with the right TSIG key will be able to perform an AXFR of the zone, regardless of the metadata values for ALLOW-AXFR-FROM for the given zone.

This can be surprising when comparing with BIND, where both IP and TSIG key-based restrictions can be implemented at the same time, on a per-zone basis.

Windows DNS:

To restrict which IP addresses are allowed to perform a zone transfer from a Windows DNS server, see the Microsoft documentation:

1. “In the DNS Manager, right-click the name of the DNS zone and click Properties.
2. On the Zone Transfers tab, click Allow zone transfer.
3. Select Only to the following servers.
4. Click Edit, then in the IP addresses of the secondary servers list, enter the IP addresses of the servers you wish to specify.
5. When you have entered all the required IP addresses, click OK.”

Alternatively, this can be done using the `dnscmd` command-line tool:

```
dnscmd ns1.example.org /zoneresetsecondaries example.org
/securelist 192.0.2.2
```

Unfortunately, Windows DNS does not support plain TSIG for restricting zone transfers as described in RFC 8945. Active Directory-enabled Windows DNS servers use another mechanism for replicating DNS zone content with each other. Another solution could be to use IPsec.

3. **Practice 3**: Zone file integrity **MUST** be controlled to avoid unexpected modifications (malicious or accidental).

   General Implementation Considerations:

   There are multiple approaches to this. For example, for static zones this could be done using the ZONEMD (RFC 8976) message digest and Resource Record, or existing revision control/versioning procedures, if those are implemented (see Host and Service security). If the zone is dynamic, and producing a message digest for the entire zone is impractical due to size or rate of change, revision control (git/svn/other) and versioning should allow auditing to narrow down when a given error or malicious change was introduced. This is assuming that zones are stored (or exported) as files before being imported into a VCS (version control system) such as Git.

   For a more general explanation of message digests in DNS as defined in RFC 8976, see this presentation.

   BIND:

   While BIND9 does correctly parse and serve ZONEMD records, the actual calculation of the hash is done with a third-party implementation. At the time of this writing, there are three known implementations mentioned in RFC 8976.

One suggested tool is Verisign's ldns-zone-digest.

See the Appendix for installation.

PowerDNS:

PowerDNS zone records typically reside in a database backend, and there is currently no provision in the PowerDNS Authoritative server to generate the ZONEMD hash. One would have to set up a secondary, pulling the zone from the PowerDNS authoritative via AXFR, and add the ZONEMD.

Windows DNS:

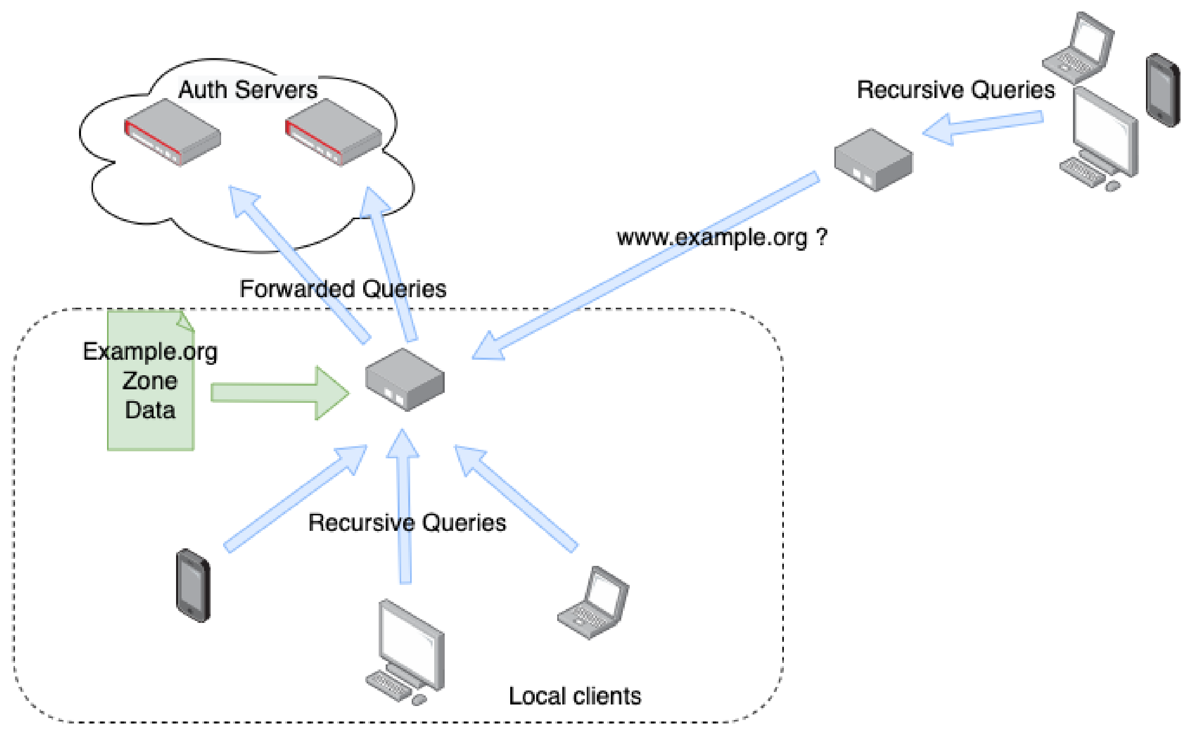
Windows DNS does not support ZONEMD.

DNS Availability and Resilience

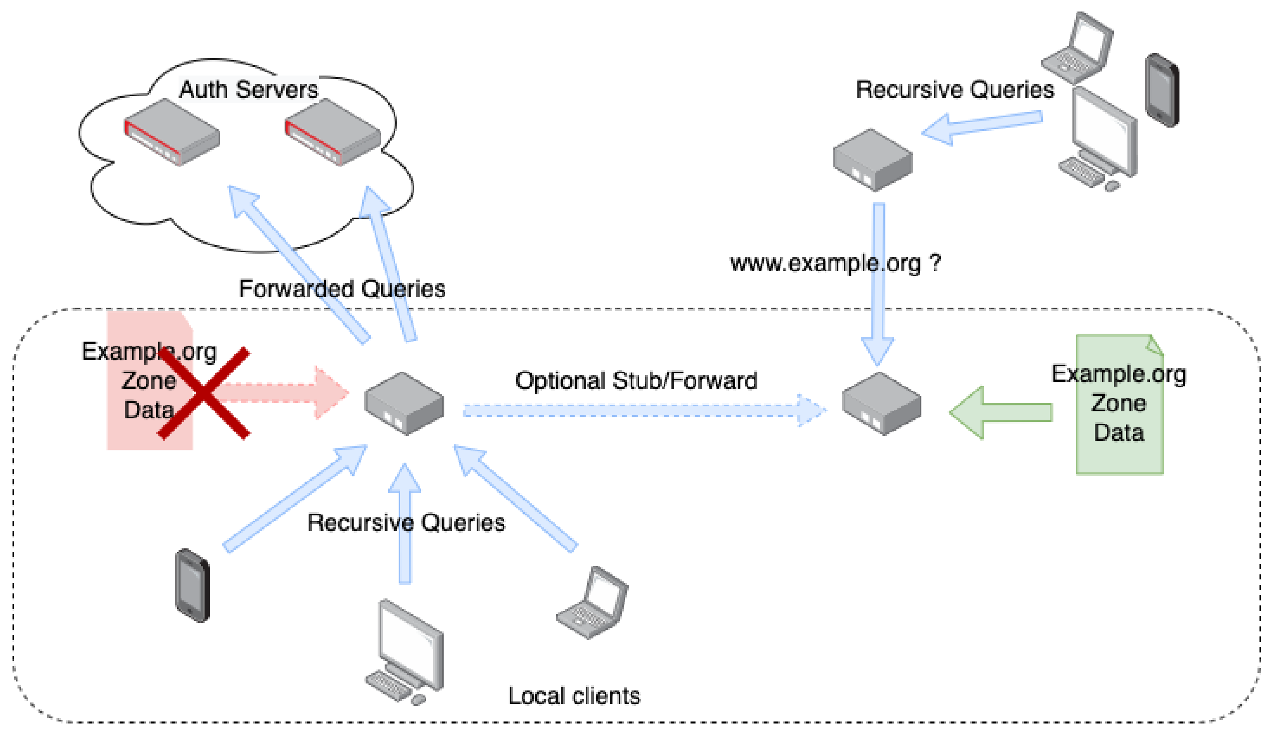
4. **Practice 4**: Authoritative and recursive DNS service **MUST NOT** coexist on the same DNS server. In the context of authoritative servers, this means you **MUST** disable recursive DNS resolution on servers configured to serve authoritative DNS data (if the software allows running both authoritative and recursive at the same time).

   General Implementation Considerations:

   Dedicated DNS recursive resolvers should be set up separately from the authoritative nameservers, as illustrated in the diagram below. Ideally, the recursive name servers will not be reachable from the wider Internet (see the corresponding BCPs on Network and Service security using ACLs), possibly on private IP space. If necessary, stub or forward type zones can be configured on the resolvers for important zones required for critical network functions and services to operate, in the event of a loss of Internet connectivity (e.g., VoIP, internal messaging/email communication).



Nameserver acting as both Recursive and Authoritative



Separation of duties - dedicated Recursive resolver and Authoritative servers

If recursion is disabled in this way, it will be necessary to set up one or more dedicated recursive DNS servers.

BIND:

Disable recursion on a BIND9 name server:

```
options {
  recursion no;
};
```

If a new, dedicated recursive name server is set up to service the clients that were previously pointing to the mixed Authoritative/Recursive server, it may be necessary to set up forward or stub zones if the zones are not publicly resolvable.

Windows DNS:

If Windows DNS was set up as part of the creation of an Active Directory domain, then all AD-integrated DNS servers will by default also offer recursive DNS resolution to clients of the domain. Public-facing servers running critical/TLD zones on Windows DNS must **not** have recursive service enabled. If any clients are configured to use these servers as resolvers, set up a dedicated recursive service for them. Stub/forward zones can be set up on the recursive service to point to the authoritative zones on the existing service if necessary.

Starting in Windows 2016, it should be possible to set up a policy restricting recursion to client subnets, according to https://docs.microsoft.com/en-us/windows-server/networking/dns/deploy/dns-policies-overview - but our team hasn't tested that.

PowerDNS:

PowerDNS Authoritative and PowerDNS Recursor are distinct products - it's not possible to offer authoritative and recursive on the same platform by design.

5. **Practice 5:** At least two distinct nameservers **MUST** be used for any given zone. Note that this is usually a requirement when registering domain names in most TLDs (gTLD, ccTLD, …).

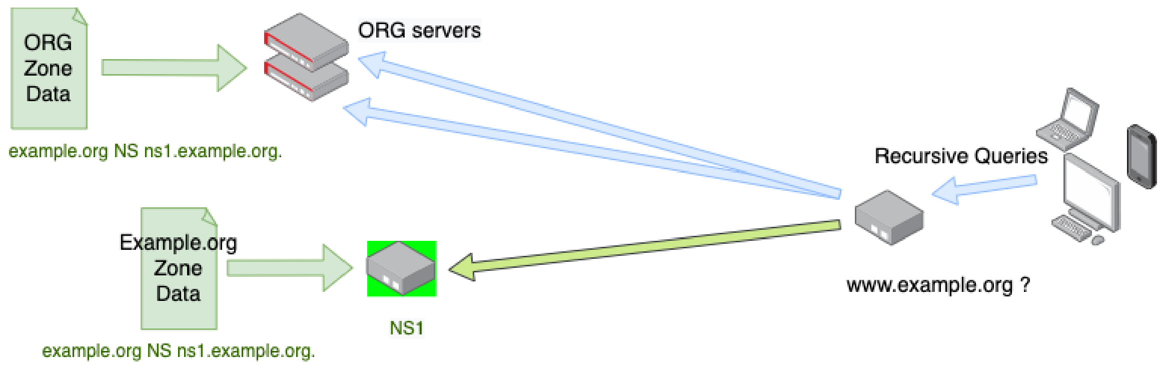
   General Implementation Considerations:

   Remember that besides setting up additional authoritative nameservers and adding them to the list of NS Resource Records in the zone, you will need to update the parent zone as well!

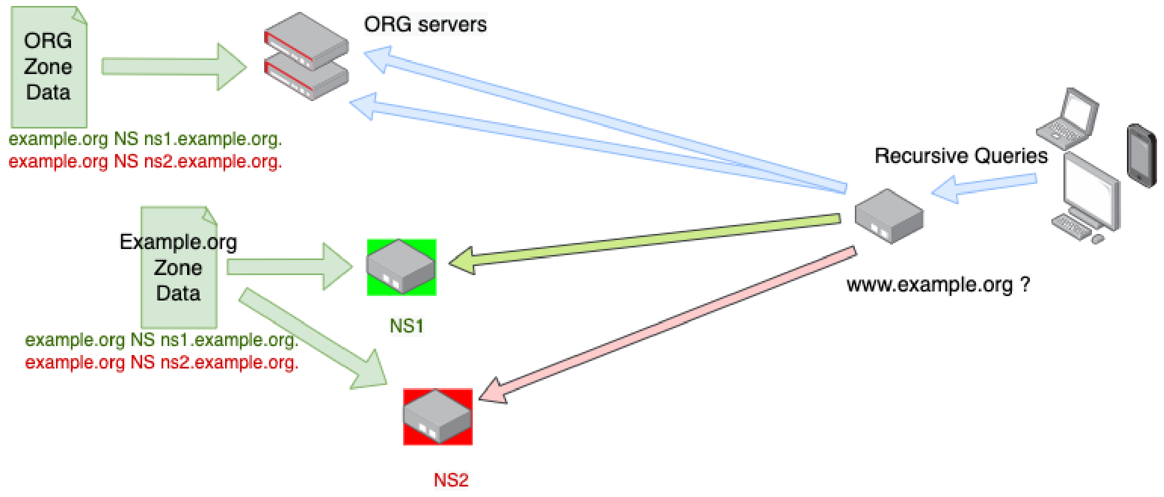
Checklist to add an additional nameserver for a zone:

1. Configure additional server (OS, DNS software)
2. Configure existing primary server(s) to allow zone transfer of zone from the new server (IP ACL, TSIG)
3. Configure new server to load the zone (or zones) from the existing primary server(s) - and verify
4. Declare the additional nameserver in the zone on the primary server: `example.org. NS nsX.example.org.`
5. Update delegation in parent zone (via registrar or registry, depending on the TLD model – or if operating a TLD, via the IANA RZM portal) to reflect the additional NS

In the figure below, notice how the additional nameserver is added both in the child zone and in the parent zone, and how recursive queries are now sent to the new server once it is published in the zone.



Single NS for zone


ORG Zone Data
ORG servers
example.org NS ns1.example.org.
example.org NS ns2.example.org.
Recursive Queries
Example.org Zone Data
www.example.org ?
NS1
example.org NS ns1.example.org.
example.org NS ns2.example.org.
NS2

Additional NS for zone

6. **Practice 6**: There **MUST** be diversity in the authoritative operations to promote resilience.

    a. **Software Diversity**: For a given zone, make sure all published nameservers aren't running the same authoritative DNS software package and version.

        General Implementation Considerations:

        Use software from two or more vendors - for instance, if running on Linux/UNIX, it could be any of the following software packages: BIND, NSD, Knot Auth, or PowerDNS. If you operate more than two nameservers, it would probably be sufficient to choose two software packages to minimize complexity, but still reduce exposure and possible downtime due to software vulnerabilities where all services are affected at the same time.

        Another solution is to use one or more third-party DNS operators to provide hosting of secondary zones. It's very likely that the contracted party (or parties) run a different software platform from the one you're already running (or at least, a different revision), thereby providing diversity.

        If it's too impractical to run different software packages, or if it is absolutely necessary to run the same version of the software package across all nameservers (say, because of feature compatibility), ensure that you have a mechanism by which you can upgrade/update the software packages while minimizing downtime. This is both for regular maintenance schedules as well as in the event of the sudden disclosure of a software vulnerability requiring immediate patching.

        One way to implement this could be using dnsdist as a load balancer, with multiple servers in the backend. It becomes possible to add/remove DNS backend servers as they are upgraded without affecting operation, even mixing different software packages. Note that while the use of load-balancing software is normally discouraged, dnsdist is built from the ground up as a high-performance, abuse-aware DNS frontend.

    b. **Network Diversity:** For a given zone, make sure all authoritative servers are not placed within the same Autonomous System (AS) or within the same subnet.

        General Implementation Considerations:

        There are multiple ways to achieve this - but the guidelines are as follows: if all nameservers are within the same subnet, even if they are placed in different physical locations (either individually routed at layer 3, or a distributed L2 network), there is a risk that a misconfiguration in a BGP configuration for the enclosing prefix would see connectivity to all nameservers be lost simultaneously.

If all nameservers are located within different subnets, at least one should be announced from a different AS to avoid similar problems as described above. This can be achieved by using a third-party operator hosting secondary zones.

c. **Geographical Diversity:** For a given zone, make sure all the authoritative servers are in different physical locations (not the same rack and room or city, region, or country).

General Implementation Considerations:

While it's obvious that having all servers (whether physical or virtual) co-located within the same rack or room doesn't allow for redundancy in case of a network or power outage, it is a good idea to aim for a different colocation facility or datacenter altogether for secondary servers - preferably in a different region or country.

7. **Practice 7:** Monitoring of the services, servers, and network equipment that make up your DNS infrastructure **MUST** be implemented.

General Implementation Considerations:

Examples of resources and services that should be monitored:

- Availability: does the DNS service answer?
    - Example: On port 53 UDP and TCP, does the DNS server return data if queried?
- Correctness: does the DNS service return the expected data?
    - Example: Query for a name and resource record type (for instance: www.example.org A, or example.org SOA), then check the result against a known good value.
- Latency: does the service respond in a timely fashion?
    - Example: How long does it take for the service to respond to the above checks? It should be within a reasonable timeframe, say, less than 5ms for most authoritative queries, and probably less than 200-300ms for recursive queries. Account for the time it takes to fetch an answer if not already in the cache, and the network latency (round trip) between the monitoring service and the DNS service you are testing.

The above three tests can be performed as a single check using most monitoring platforms/services.

Example with the Nagios plugin, check_dig:

```
check_dig -4 -H a.icann-servers.net -l www.icann.org -w 2 -c 5 -a \
    www.vip.icann.org
```

The queried server is a.icann-servers.net, using IPv4 (‘-4’). The queried name is ‘www.icann.org’. The settings -w 2 and -c 5 set a warning and critical threshold if the server hasn’t responded before 2 and 5 seconds have elapsed, respectively. `-a` is the expected result, in this case ‘www.vip.icann.org’. We don’t set a queried record type (-T), so it defaults to A.

Example output:

```
DNS OK - 0.123 seconds response time (www.icann.org.  3600 IN CNAME
www.vip.icann.org.)|time=0.122713s;2.000000;5.000000;0.000000
```

- Synchronization: i=Is the zone data identical across all name server instances (SOA/serial check, for example)?
    - Example: One could implement a check using check_dig to ensure that all nameservers for a given zone have the same serial number. Alternatively, one could use a third-party plugin such as “check-dns-serial”, available at https://exchange.nagios.org/directory/Plugins/Network-Protocols/DNS/Check-DNS-Serial/details

The same techniques apply for monitoring the availability and reachability of intermediate network devices - ICMP checks are often enough in most cases to detect failure of a router or switch in front of a DNS server.

If monitoring with a third-party service, there are many online providers which provide remote monitoring of availability and reachability.