



# Managing DNS Server Properties

This chapter explains how to set the DNS server parameters. Before you proceed with the tasks in this chapter, read [Chapter 14, “Managing Zones,”](#) which explains how to set up the basic properties of a primary and secondary zone.

## Managing DNS Servers

You can manage the DNS server, including viewing its health, statistics, and logs; starting, stopping, and reloading it; running certain commands; and editing the server attributes.

You can view the server status and health, and stop, start, and reload the server. In the local cluster Web UI, click **DNS**, then **DNS Server** to open the Manage DNS Server page.

## Running DNS Commands

Access the commands by using the Run icon (▶) in the Commands column to open the DNS Server Commands page. Each command has its own Run icon (click it, then **Return** when finished):

- Flush cache—See the [“Flushing DNS Cache”](#) section on page 16-12.
- Force all zone transfers—See the [“Enabling Zone Transfers”](#) section on page 14-13.
- Scavenge all zones—See the [“Scavenging Dynamic Records”](#) section on page 27-12.
- Remove non-authoritative RR set—See the [“Removing Cached Records”](#) section on page 15-4.



**Note**

If you find a server error, investigate the server log file for a configuration error, correct the error, return to this page, and refresh the page.

## Configuring DNS Server Network Interfaces

You can configure the network interfaces for the DNS server from the Manage Servers page in the Web UI at the local cluster. To get there, click **Servers**, then **Manage Servers**. Click the Interfaces icon (🖨️) for the DNS server to open the Manage DNS Server Network Interfaces page.

This page shows the available network interfaces that you can configure for the server. By default, the server uses all of them. To configure an interface, click the Configure icon (✎) in the Configure column for the interface. This adds the interface to the Configured Interfaces table, where you can edit or delete

it. Clicking the name of the configured interface opens the Edit DNS Server Network Interface page, where you can change the address and port of the interface. Click **Modify Interface** when you are done editing, then click **Return** to return to the Manage Servers page.

**Note**

The IPv6 functionality in DNS requires IPv4 interfaces to be configured except if the DNS server is isolated and standalone (it is its own root and is authoritative for all queries).

## Setting DNS Server Properties

You can set properties for the DNS server itself, along with those you already set for its zones.

### Setting General Server Properties

You can display DNS general server properties, such as the name of the server's cluster or host machine and the version number of the Network Registrar DNS server software. You can change the internal name of the DNS server by deleting the current name and entering a new one. This name is used for notation and does not reflect the server's official name. Network Registrar uses the server's IP address for official name lookups and for DNS updates (see [Chapter 27, "Configuring DNS Update"](#)).

- 
- Step 1** In the local cluster Web UI, click **DNS**, then **DNS Server** to open the Manage DNS Server page.
- Step 2** Click the name of the server to open the Edit DNS Server page. The page displays all the DNS server attributes.
- In the CLI, use **dns [show]** to display the DNS server properties.
- Step 3** Click **Modify Server** to save the DNS server attribute modifications.
- 

### Defining Forwarders for Servers

Sites that must limit their network traffic for security reasons can designate one or more servers to be forwarders that handle all off-site requests before the local server goes out to the Internet. Over time, the forwarders build up a rich data cache that can satisfy most requests.

Forwarders are useful in that they:

- Reduce the load on the Internet connection—Forwarders build up a cache and thus reduce the number of requests sent to external nameservers and improve DNS performance.
- Improve the DNS response to repeated queries—The forwarder's cache can answer most queries.
- Handle firewalls—Hosts that do not have access to root nameservers can send requests to the forwarder that does.

**Tip**

Restrict the nameserver even more by stopping it from even attempting to contact an off-site server. This kind of server uses forwarders exclusively. It answers queries from its authoritative and cached data, but it relies completely on the forwarders for data not in its cache. If the forwarder does not provide an answer, then the slave mode server does not try to contact other servers.

You can have multiple forwarders configured. If the first forwarder does not respond after a default of eight seconds, Network Registrar asks each remaining forwarder in sequence until one answers, or until it gets to the end of the list. If the DNS server does not get an answer, the next step depends on whether you have slave mode on or off:

- If slave mode is on, the DNS server stops searching and responds that it cannot find the answer.
- If slave mode is off, the DNS server sends the query to the domain's designated nameservers as if there were no forwarders listed.

You can modify the eight-second default for the forwarding retry interval through the DNS *forward-retry-time* attribute. Because these queries are recursive and can require more time for the forwarder to resolve, set this interval to the *request-expiration-time* (90 seconds by default) divided by one fewer than the total number of configured forwarders or resolution exception servers. (See also the “[Enabling Recursive Queries](#)” section on page 16-6.)

In the local cluster Web UI, on the Edit DNS Server page, under the Forwarders category, enter the IP addresses of the forwarding servers, click whether you want slave mode enabled or disabled, then click **Modify Server**.

In the CLI:

- To specify the address (or space-separated addresses) of nameservers to use as forwarders, use **dns addForwarder**.
- To designate the server as a slave, use **dns enable slave-mode**.
- To list the current forwarders, use **dns listForwarders**.
- To edit your forwarder list, you must remove any offending forwarder and reenter it.
- To remove a forwarder or list of forwarders, use **dns removeForwarder**.

**Note**

---

Subzone forwarding (see the “[Setting Subzone Forwarding](#)” section on page 16-3) and resolution exception (see the “[Using Resolution Exception](#)” section on page 16-4) override any forwarders you set.

---

## Setting Subzone Forwarding

For zones with forwarders set (see the “[Defining Forwarders for Servers](#)” section on page 16-2), the normal Network Registrar behavior is to ignore delegation to subzone nameservers and forward queries to these forwarding servers. You would normally need to set a resolution exception (see the “[Using Resolution Exception](#)” section) to the subzone server. This might be impractical for large numbers of subzones. With the zone's *subzone-forward* attribute set to **no-forward**, when the server receives a query for any of its subzones, it tries to find relevant subzone NS records, resolve their corresponding IP addresses, and delegate the query to those IP addresses. The default is to use the forwarders, if requested. However, subzone forwarding overrides any requested forwarders.

For example, with the example.com zone and its subzones boston.example.com and chicago.example.com, if you set the no-forward option:

```
nrcmd> zone example.com set subzone-forward=no-forward
```

Queries do not go to the forwarder, but their NS and corresponding A records are found. The default is *subzone-forward=normal*, which causes queries to be forwarded.

## Using Resolution Exception

If you do not want the DNS servers to use the standard resolution method to query the root nameserver for certain names outside its domain, use resolution exception. This bypasses the root nameservers and targets a specific server to handle name resolution.

For example, example.com has four subsidiaries: Red, Blue, Yellow, and Green. Each has its own domain under the .com domain. When users at Red want to access resources at Blue, their DNS server knows that it is not authoritative for Blue and appeals to the root nameservers. These queries cause unnecessary traffic, and in some cases fail because internal resources are often barred from external queries or sites that use unreachable private networks without unique addresses.

Resolution exception solves these problems. Red's administrator lists all the other example.com domains that users might want to reach and at least one corresponding nameserver. When a Red user wants to reach a Blue server, the Red server asks the Blue server instead of querying the root.



### Note

When resolution exception is configured for a subzone, the resolution exception server is not queried; the subzone NS records are used instead. With no resolution exception defined, when global forwarding is set (see the “[Setting Subzone Forwarding](#)” section), any query for the subzone delegation goes to the forwarder, and not to the server that is authoritative for that subzone. However, if you set the *subzone-forward* attribute to *no-forward* for a zone, the zone's server is set as the implicit resolution exception server for all the subzones, and any forwarding and explicitly set resolution exceptions are ignored for that zone. Even though defining a resolution exception at a subzone name is a useful way to prevent queries to a specific subzone from being forwarded to the global forwarder, to do this for all subzones of a zone, use subzone forwarding.

In the local cluster Web UI, on the Edit DNS Server page, under the Resolution Exceptions category, enter the domain name and IP address or addresses of each excepted nameserver, then click **Modify Server**. To delete an exception list, click the Delete icon (🗑️) next to the domain name and IP address or addresses of the nameserver you want to remove, then click **Modify Server**.

In the CLI:

- To add the exception domains and servers, separated by spaces, use **dns listExceptions** and **dns addException**. Use this command only if you do not want your DNS server to use the standard name resolution for names outside the local authoritative zone.
- To remove a resolution exception, use **dns removeException**.
- To replace a resolution exception, use **dns addException** with the new values. You must also flush the cache so that the server does not refer to the old resolution values in cache.

## Configuring Caching-Only Servers

By definition, all servers are caching servers, because they save the data that they receive until it expires. However, you can create a caching-only server that is not authoritative for any zone. This type of server's only function is to answer queries by storing in its memory data from authoritative servers. The caching-only server can then learn or cache the data to answer subsequent queries. Caching query responses considerably reduces the overhead (and latency) of subsequent queries to these cached records

When you first install Network Registrar, the DNS server automatically becomes a non-authoritative, caching-only server until you configure zones for it. If you keep the DNS server as a caching-only server, you must have another primary or secondary DNS server somewhere that is authoritative and to which the caching-only server can refer. The caching-only server should never be listed as an authoritative server for a zone.

You must set up a caching-only server to respond to recursive queries, where a server keeps trying to get to an authoritative server so that it can update its cache with the address resolution data. Network Registrar servers are recursive by default.

To tune the performance of a caching-only server:

- Increase the in-memory cache to the maximum possible value that is aligned to the operating system's physical memory capacity by setting the *mem-cache-size* value:

```
nrcmd> dns set mem-cache-size=200MB
```

- When using a large cache, you might want to disable persisting the cache, so that reload time is not impacting the need to save the large cache. Disable the *persist-mem-cache* attribute:

```
nrcmd> dns disable persist-mem-cache
```

(However, note that disabling the attribute discards the cache at each server reload.)

- Restrict queries to certain clients only by setting the *restrict-query-acl* attribute:

```
nrcmd> dns set restrict-query-acl=myaccesslist
```

In the local cluster Web UI, click **DNS, DNS Server**, then the name of the server to open the Edit DNS Server page. Under the Forwarders category, set the previously described attributes.

In the CLI, use the previously listed commands.

## Specifying Delegation-Only Zones

You can instruct the server to expect only referrals when querying the specified zone. In other words, you want the zone to contain only NS records, such as for subzone delegation, along with the zone's apex SOA record. This can filter out "wildcard" or "synthesized" data from authoritative nameservers whose undelegated (in-zone) data is of no interest. Enable the DNS server's *delegation-only-domains* attribute for this purpose.

## Defining Root Nameservers

Root nameservers know the addresses of the authoritative nameservers for all the top-level domains. When you first start a newly installed Network Registrar DNS server, it uses a set of preconfigured root servers, called root hints, as authorities to ask for the current root nameservers.

When Network Registrar gets a response to a root server query, it caches it and refers to the root hint list. When the cache expires, the server repeats the process. Because Network Registrar has a persistent cache, it does not need to requery this data when it restarts. The time to live (TTL) on the official root server records is currently six days, so Network Registrar requeries every six days, unless you specify a lower maximum cache TTL value (see the ["Setting Maximum Cache TTLs"](#) section on page 16-11).

Because the configured servers are only hints, they do not need to be a complete set. You should periodically (every month to six months) look up the root servers to see if the information needs to be altered or augmented.

In the local cluster Web UI, on the Edit DNS Server page, under the Root Nameservers category, enter the domain name and IP address of each additional root nameserver, then click **Modify Server**.

In the CLI, use **dns addRootHint** (see the Usage Guidelines for the **dns** command in the *Cisco CNS Network Registrar CLI Reference Guide*).

## Enabling Recursive Queries

There are two types of queries—*recursive* and *iterative* (nonrecursive). DNS clients typically generate recursive queries, where the nameserver asks other DNS servers for any non-authoritative data not in its own cache. With an iterative query, the nameserver answers the query if it is authoritative for the zone, has the answer in its cache, or tells the client which nameserver to ask next. You often want to make a root server iterative instead of recursive. Recursion is like saying, “Let me talk to Bob and get back to you.” Iteration is like saying, “Let me direct you to Bob for the information.”

You can also restrict the clients from which to honor recursive queries by setting the *restrict-recursion-acl* attribute to an access control list (ACL); see the [“Restricting Zone Queries” section on page 16-8](#).

Here is a list of other attributes that you can set to optimize recursive query performance, although you would generally leave these values to their defaults:

- *forward-retry-time*—Retry interval for forwarding queries to forwarders or resolution exception servers (default 8 seconds); see the [“Defining Forwarders for Servers” section on page 16-2](#).
- *request-expiration-time*—Expiration time of general DNS queries (default 90 seconds).
- *request-retry-time*—Retry time interval for general DNS queries (default 4 seconds).
- *tcp-query-retry-time*—Retry time for DNS queries over TCP, in response to truncated UDP packets (default 10 seconds).

## Enabling Round-Robin

A query might return multiple A records for a nameserver. To compensate for most DNS clients starting with, and limiting their use to, the first record in the list, you can enable *round-robin* to share the load. This method ensures that successive clients resolving the same name will connect to different addresses on a revolving basis. The DNS server then rearranges the order of the records each time it is queried. It is a method of load sharing, rather than load balancing, which is based on the actual load on the server.



### Tip

---

Adjust the switchover rate from one round-robin server to another using the TTL property of the server’s A record.

---

In the local cluster Web UI, on the Edit DNS Server page, under the Miscellaneous Options and Settings category, set the *Enable round-robin* attribute to enabled, then click **Modify Server**.

In the CLI, use **dns get round-robin** to see if round-robin is enabled (it is by default). If not, use **dns enable round-robin**.

## Enabling Subnet Sorting

If you enable subnet sorting, as implemented in BIND 4.9.7, the Network Registrar DNS server confirms the client's network address before responding to a query. If the client, server, and target of the query are on the same subnet, and the target has multiple A records, the server tries to reorder the A records in the response by putting the target's closest address first in the response packet. DNS servers always return all of a target's addresses, but most clients use the first address and ignore the others.

If the client, DNS server, and target of the query are on the same subnet, Network Registrar first applies round-robin sorting and then applies subnet sorting. The result is that if you have a local answer, it remains at the top of the list, and if you have multiple local A records, the server cycles through them.

In the local cluster Web UI, on the Edit DNS Server page, under the Advanced Options and Settings category, set the *Enable subnet sorting* attribute to enabled, then click **Modify Server**.

In the CLI, use **dns enable subnet-sorting** or **dns disable subnet-sorting** (the default).

## Enabling Incremental Zone Transfers (IXFR)

Incremental Zone Transfer (IXFR, described in RFC 1995) allows only changed data to be transferred between servers. This is especially useful in dynamic environments. IXFR works together with NOTIFY (see the “[Enabling NOTIFY](#)” section on page 16-8) to ensure more efficient zone updates.

Primary zone servers always provide IXFR. Explicitly enabling IXFR then provides this function for secondary zone servers, so that it is meaningless to enable it unless a server has no secondary zones. The difference between this global setting and that for the secondary zone is that the global setting is the default value if a secondary zone has no specific setting.

- 
- Step 1** In the local cluster Web UI, on the Edit DNS Server page, under the Zone Defaults category, set the *Request incremental transfers (IXFR)* attribute to enabled.
- Step 2** For a secondary zone, you can also fine tune the incremental zone transfers by setting the *Maximum IXFR-only interval (secondary zones)* attribute. This is the longest interval to run on IXFRs alone before forcing a full zone transfer (AXFR), usually set to a period such 24 hours or a few days.
- In the CLI, use **dns enable ixfr-enable**. By default, the *ixfr-enable* attribute is enabled (see the **dns** command in the *Network Registrar CLI Reference*).
- Step 3** Click **Modify Server**.
- 

## Change Sets and Checkpointing

Network Registrar maintains a change set database that collects recent changes to RR data before they are committed to the authoritative database (auth.db). The DNS server checks the change set database first before it checks the auth.db when responding to or performing:

- External DNS updates.
- Full or incremental zone transfers.
- Zone checkpointing (see later in this section).
- Stale record scavenging (see the “[Scavenging Dynamic Records](#)” section on page 27-12).
- Incremental or full configuration database reloads.

A single DNS change set can consist of one or more RRs. The DNS server periodically flushes the data to a transaction log in the database, where it accumulates before being committed to the `auth.db`. The change set database is backed up during the usual `mcdshadow` backup.

To keep the change set database from growing without bounds, the server trims it periodically. The effect of this is, for example, if a DNS client requests a zone IXFR based on a serial number for RRs that were trimmed, the DNS server cannot perform an IXFR, but must perform a full zone transfer (AXFR).

Zone checkpointing creates a snapshot of the zone data that may be necessary to recreate the `auth.db` in case the DNS server is interrupted. The server automatically updates the checkpoint files periodically. In addition to occurring with each change set that is over a certain size threshold, zone checkpointing happens by default every three hours. You can adjust this interval, from between one and 168 hours, using the `checkpoint-interval` DNS server or zone attribute.

In the local Web UI, you can manually force a zone checkpoint. Click the Run icon (▶) on the List/Add Zones page or List/Add Reverse Zones page. Then, on the Zone Commands for Zone page, click the Run icon (▶) next to Checkpoint zone.

In the CLI, use `zone name chkpt`, or dump the zone checkpoint file in a more humanly-readable form by using `zone name dumpchkpt`.

## Restricting Zone Queries

You can restrict clients to query only certain zones based on an access control list (ACL). An ACL can contain source IP addresses, network addresses, TSIG keys (see the “[Transaction Security](#)” section on page 27-6), or other ACLs. The ACL set by the zone’s `restrict-query-acl` attribute serves as a filter for non-authoritative queries. If a query targets an authoritative zone, the zone’s ACL applies.

You can also restrict the server to honor recursive requests from certain clients only by setting the `restrict-recursion-acl` attribute to an ACL that includes these clients (see “[Access Control Lists](#)” section on page 27-5). If you unset `restrict-recursion-acl`, anyone can request a recursive query.

## Enabling NOTIFY

The NOTIFY protocol, described in RFC 1996, lets the Network Registrar DNS primary server inform its secondaries that zone changes occurred. The NOTIFY packet does not indicate the changes themselves, just that they occurred, and this triggers a zone transfer request. Use NOTIFY in environments where the namespace is relatively dynamic.

Because a zone’s master server cannot know specifically which secondary server transfers from it, Network Registrar notifies all nameservers listed in the zone’s NS records. The only exception is the server named in the SOA primary master field.

You can use IXFR and NOTIFY together, but this is not necessary. You can disable NOTIFY for a quickly changing zone for which immediate updates on all secondaries does not warrant the constant NOTIFY traffic. Such a zone might benefit from having a short refresh time and a disabled NOTIFY.

- 
- Step 1** In the local cluster Web UI, on the Edit DNS Server page, under the Zone Defaults category, set the *Send zone change notification (NOTIFY)* attribute to enabled.
  - Step 2** Set any of the other NOTIFY attributes.
  - Step 3** Click **Modify Server**.
  - Step 4** To add nameservers in addition to those specified in NS records, click **Zones**.

- Step 5** Click the zone name.
- Step 6** Add a comma-separated list of servers using the *notify-set* attribute on the Edit Zone page.
- Step 7** Set the *notify* attribute to true.

In the CLI, use **dns enable notify**. NOTIFY is enabled by default. You can also enable NOTIFY at the zone level, where you can use **zone name set notify-set** to specify an additional comma-separated list of servers to notify beyond those specified in NS records.

- Step 8** Click **Modify Zone** on that page.
- 

## Setting Advanced Server Properties

You can set these advanced server properties:

- SOA and secondary server attributes
- Prefetch glue records
- Report lame delegation
- Enable relaxed DNS update
- Set cache time limits and size
- Set local and external port numbers
- Handle rogue A record queries
- Set debug

## Setting SOA Time to Live

The SOA record's time to live (TTL) is usually determined by the zone's default TTL. However, you can explicitly set the SOA TTL, which sets the maximum number of seconds a server can cache the SOA record data. For example, if the SOA TTL is set for 3600 seconds (one hour), an external server must remove the SOA record from its cache after an hour and then query your nameserver again.

Network Registrar responds to authoritative queries with an explicit TTL value. If there is no explicit TTL value, it uses the default TTL for the zone, as set by the value of the *defttl* zone attribute. Databases originating from versions of Network Registrar earlier than Release 3.5 do not have the *defttl* zone attribute, and use the minimum TTL in the zone's SOA record for the default TTL.

Normally, Network Registrar assumes the default TTL when responding with a zone transfer with RRs that do not have explicit TTL values. If the default TTL value for the zone is administratively altered, Network Registrar automatically forces a full zone transfer to any secondary DNS server requesting a zone transfer.

- 
- Step 1** In the local cluster Web UI, on the Add Zone or Edit Zone page, set the Zone Default TTL, which defaults to 24 hours.
- Step 2** If you wish, set the SOA TTL, which is the TTL for the SOA records only. It defaults to the Zone Default TTL value.
- Step 3** You can also set a TTL value specifically for the NS records of the zone. Set the *nsttl* value listed under the attributes. This value also defaults to the Zone Default TTL value.

In the CLI, use **zone *name* set defttl**.

**Step 4** Click **Modify Zone**.

**Step 5** Reload the server.

---

## Setting Secondary Refresh Times

The secondary refresh time is how often a secondary server communicates with its primary about the potential need for a zone transfer. A good range is from an hour to a day, depending on how often you expect to change zone data.

If you use NOTIFY, you can set the refresh time to a larger value without causing long delays between transfers, because NOTIFY forces the secondary servers to notice when the primary data changes. For details about NOTIFY, see the [“Enabling NOTIFY” section on page 16-8](#).

In the local cluster Web UI, on the Add Zone or Edit Zone page, set the Secondary Refresh field to the refresh time, which defaults to three hours. Make any other changes, then click **Modify Zone**.

In the CLI, use **zone *name* set refresh**. The default is 10800 seconds (three hours).

## Setting Secondary Retry Times

The DNS server uses the secondary retry time between successive failures of a zone transfer. If the refresh interval expires and an attempt to poll for a zone transfer fails, the server continues to retry until it succeeds. A good value is between one-third and one-tenth of the refresh time. The default is one hour.

In the local cluster Web UI, on the Add Zone or Edit Zone page, set the Secondary Retry field to the retry time, which defaults to one hour. Make any other changes, then click **Modify Zone**.

In the CLI, use **zone *name* set retry**.

## Setting Secondary Expiration Times

The secondary expiration time is the longest time a secondary server can claim authority for zone data when responding to queries after it cannot receive zone updates during a zone transfer. Set this to a large number that provides enough time to survive extended primary server failure. The default is seven days.

In the local cluster Web UI, on the Add Zone or Edit Zone page, set the Secondary Expire field to the expiration time, which defaults to seven days. Make any other changes, then click **Modify Zone**.

In the CLI, use **zone *name* set expire**.

## Fetching Glue Records

A glue record is a DNS A record that specifies the address of a zone’s or subzone’s authoritative nameserver. It is an informational record in a query response. For example, most answers include NS records, which then cause the inclusion of A records to resolve the NS record name into an address. These A records are the glue records. Disabling the *no-fetch-glue* attribute tells the server to find records that it would not normally find, so that it can include them in answers to subsequent queries.

In the local cluster Web UI, on the Edit DNS Server page, ensure that *Don't fetch missing glue records* is set to disabled.

In the CLI, use **dns disable no-fetch-glue** (the default).

## Reporting Lame Delegation

Lame delegation occurs when a DNS server listed in the parent's delegation of a zone does not know that it is authoritative for the zone. The server can detect and report this when, in the process of tracking down an answer, it is referred to a server that in turn refers to another server for a domain closer to the root (actually farther from the answer). Lame delegation does not indicate a problem with the DNS configuration, but with the configuration at the DNS server you are querying. You cannot do anything to correct lame delegation at other domains, unless you happen to be authoritative for the domain as well.

Note that setting the *Report lame delegations (lame-deleg-notify)* attribute has the same effect as setting the DNS log setting *lame-delegation*.

In the local cluster Web UI, on the Edit DNS Server page, ensure that *Report lame delegations* is set to enabled.

In the CLI, use **dns enable lame-deleg-notify** (the default).

## Setting Maximum Negative Cache Times

To ensure a quick response to repeated requests for the same information, the DNS server maintains a cache of data it learns from other servers on behalf of its clients. This includes negative information, such as “no such name” or “no such data,” as specified by RFC 2308. It is important to discard this information at some point to accommodate namespace changes at the authoritative source.

The maximum negative cache time establishes an upper bound on the time a negative cache entry can be valid. This value should be obtained from the SOA record in the negative response. If the originating zone has an abnormally large TTL value, the maximum negative cache time value reduces that value, limiting the time that the entry remains negatively cached. Choose this value carefully to balance the need to eliminate long negative cache entries with the desire to cache negative answers meaningfully.

Note that you can effectively reduce the maximum negative cache time through the maximum cache TTL value, because this value applies to all cache entries, positive and negative. See the [“Setting Maximum Cache TTLs”](#) section. The smaller of the TTL values of the *max-cache-ttl* and *max-negcache-ttl* wins when caching negative entries.

In the local cluster Web UI, on the Edit DNS Server page, set the *Max. negative answer caching TTL* value; the default is one hour.

In the CLI, use **dns set max-negcache-ttl** (the default is 60 minutes).

## Setting Maximum Cache TTLs

The Maximum Cache TTL property specifies the maximum time that you want Network Registrar to retain cached information. TTL is the amount of time that any nameserver is allowed to cache data learned from other nameservers. Each record added to the cache arrives with some TTL value. When the TTL period expires, the server must discard the cached data and get new data from the authoritative nameservers the next time it sends a query. This parameter limits the lifetime of records in the cache whose TTL values are very large. The default value is seven days (604800 seconds).

In the Web UI, on the Edit DNS Server page, set the *Max. resource record caching TTL* value; the default is one week.

In the CLI, use **dns set max-cache-ttl**.

## Setting Maximum Memory Cache Sizes

The maximum memory cache size property specifies how much memory space you want to reserve for the DNS in-memory cache. The more memory that you allocate for the cache, the less frequently the server uses disk cache. The default is 200 KB. One entry is approximately 100 bytes.

In the local cluster Web UI, on the Edit DNS Server page, set the *Max. memory cache size* value; the default is 200 KB.

In the CLI, use **dns set mem-cache-size**.

## Flushing DNS Cache

The Network Registrar cache flushing function lets you stop the disk cache file from growing. However, the actual behavior depends on whether the DNS server is running or stopped. If you flush the cache:

- While the server is running, Network Registrar clears all expendable entries from the cache database file. Flushing the cache does not shrink the file because of the nature of the database, but does create free space in it. Because the memory cache is unaffected by this operation, recently used cache entries are not lost, and performance is not significantly affected.
- With the server stopped, Network Registrar interprets the request to flush all entries and removes the cache file. It then reinitializes the database when you restart the server.

To clear a cache that grew too large, or when changing a resolution exception, stop the server, enter the command, then restart the server. Stopping the server does not terminate the server process, but stops it from handling further requests. For details on resolution exception, see the [“Using Resolution Exception” section on page 16-4](#).

In the local cluster Web UI, on the Manage DNS Server page, click the Run icon (▶) in the Commands column to open the DNS Server Commands page (see [Figure 6-3 on page 6-2](#)). On this page, click the Run icon (▶) next to Flush cache. If you want to flush cache older than a certain time, enter the time in the Time field next to Flush cache older than, then click the Run icon.

In the CLI, use **dns flushCache**.

## Handling Rogue Address Records and Other Cache Attributes

You may become victim of a suspicious denial-of-service attack where a rogue host targets Address (A) RR queries to a caching DNS server. These A record queries contain names that resemble IP addresses. To avoid overloading the DNS server’s cache.db file with negative responses from the root, the server no longer tries to resolve these queries. The *fake-ip-name-response* DNS attribute (*Fake Responses for IP address-like names* in the Web UI) is enabled by default to effect this. When the server receives a query for a non-authoritative name, it consults its cached data. If the server cannot answer from cached data, it examines the queried name to see if it resembles an IP address (four decimals each separated by a dot with no trailing or preceding characters). If so, it does not forward the query. Instead, the server sends a response with a NAME ERROR (NXDOMAIN) return code, and does not cache this synthesized negative response.

The server acts on the *save-negative-cache-entries* (*Save negative cache entries to disk* in the Web UI) and *cache-write-ttl-threshold* attributes when it evicts entries from its memory cache to the *cache.db* file. It typically evicts positive and negative query responses from in-memory cache when the in-memory cache is full and the server needs to insert a new entry. The server evicts the least-recently-used entry. If you disable *save-negative-cache-entries*, the server does not store evicted negative entries in the *cache.db* file, but simply discards them from the in-memory server cache. If the *cache-write-ttl-threshold* value is non-zero (it is zero by default), the server only persists entries from the in-memory cache to the *cache.db* file if the entry's TTLs are greater than this value. Otherwise, the server discards the (soon to be, but not yet, expired) RR. A zero value causes the server to always persist unexpired RRs.

In the local cluster Web UI, on the Edit DNS Server page, set *Fake responses for IP address-like names* to enabled (the default) and *Save negative cache entries to disk* to disabled (it is enabled by default).

In the CLI, enable the *fake-ip-name-response* and *save-negative-cache-entries* attributes, respectively.

## Setting Query Source Addresses and Port Numbers

The query source address is the source IP address from which the DNS server should send queries to other servers when resolving names for clients. A value of 0.0.0.0 indicates that the best local address is used based on the destination.

The query source port is the UDP port number from which the DNS server should send queries to other servers when resolving names for clients. A value of zero indicates a random port, which is often used when the server is behind a firewall, and an unset value causes it to be sent from the local port (see the [“Setting Local and External Port Numbers”](#) section).

In the local cluster Web UI, on the Edit DNS Server page, set *Query source IP address* and *Query source UDP port* attributes.

In the CLI, use **dns set query-source-address** and **dns set query-source-port**.

## Setting Local and External Port Numbers

If you are experimenting with a new group of nameservers, you might want to use nonstandard ports for answering requests and asking for remote data. The local port and external port settings control the TCP and UDP ports on which the server listens for name resolution requests, and to which port it connects when making requests to other nameservers. The standard value for both is port 53. If you change these values during normal operation, the server will appear to be unavailable.

In the local cluster Web UI, on the Edit DNS Server page, set *Listening port* and *Remote DNS servers port* to different ports than the default value of 53.

In the CLI, use **dns set local-port-num** and **dns set remote-port-num**.

## Tuning DNS Properties

Here are some suggestions to tune some of the DNS server properties:

- The *Notify send min. interval* DNS server attribute (*notify-min-interval* in the CLI)—Minimum interval required before sending notification of consecutive changes on the same zone to a server. The default is two seconds. For very large zones, you might want to increase this value to exceed the maximum time to send an outbound full zone transfer. This is recommended for secondary servers that receive inbound incremental zone transfers and send out full transfers to other secondaries. These include older BIND servers that do not support incremental zone transfers. Inbound incremental transfers may abort outbound full transfers.
- The *Notify delay between servers* DNS server attribute (*notify-send-stagger* in the CLI)—Interval to stagger notification of multiple servers of a change. The default is one second, but you may want to raise it to up to five seconds if you need to support a large number of zone transfers distributed to multiple servers.
- The *Notify wait for more changes* DNS server attribute (*notify-wait* in the CLI)—Time to delay, after an initial zone change, before sending change notification to other nameservers. The default is five seconds, but you may want to raise it to 15, for the same reason as given for the *notify-min-interval* attribute.
- The *Max. memory cache size* DNS server attribute (*mem-cache-size* in the CLI)—Size of the in-memory record cache, in kilobytes. The default is 200 KB, but you may want to raise it to 1000 KB for larger networks.
- The *Report lame delegations* DNS server attribute (*lame-deleg-notify* in the CLI)—Network Registrar should notice and log when a DNS server listed in a parent-zone's delegation of subzones does not know that it is authoritative for the zone. This is normally disabled, but you may want to enable it. This attribute has the same effect as setting the log settings to *lame-delegation*.
- The IXFR check box in the Foreign Servers section of the Edit DNS Server page, or **remote-dns address/mask create ixfr** in the CLI—Adding an entry for a server or group of servers allows controlling whether or not IXFR should occur when doing zone transfers from those servers.

## Troubleshooting DNS Servers

Useful troubleshooting hints tools to diagnose the DNS server include:

- Restoring a loopback zone—A loopback zone is a reverse zone that enables a host to resolve the loopback address (127.0.0.1) to the name *localhost*. The loopback address is used by the host to enable it to direct network traffic to itself. You can configure a loopback zone manually or you can import it from an existing BIND zone file. (See the Usage Guidelines for the **zone** command in the *Network Registrar CLI Reference*.)
- Listing the values of the DNS server properties—In the Web UI, click **DNS**, then **DNS Server** to open the Edit DNS Server page. In the CLI, use **dns show**.
- Choosing from the DNS log settings to give you greater control over existing log messages—Use the *Log settings* attribute on the Edit DNS Server page in the Web UI, or **dns set log-settings** in the CLI with one or more of these keyword or numeric values, separated by commas (see [Table 16-1](#)). Restart the server if you make any changes to the log settings.

**Table 16-1 DNS Log Settings**

<b>Log Setting (Numeric Equivalent)</b>	<b>Description</b>
config (1)	Server configuration and deinitialization.
ddns (2)	High level dynamic update messages.
xfr-in (3)	Inbound full and incremental zone transfers.
xfr-out (4)	Outbound full and incremental zone transfers.
notify (5)	NOTIFY transactions.
datastore (8)	Data store processing that provides insight into various events in the server's embedded databases.
scavenge (9)	Scavenging of dynamic RRs (see the <a href="#">“Scavenging Dynamic Records”</a> section on page 27-12).
scavenge-details (10)	More detailed scavenging output (disabled by default).
server-operations (11)	General high-level server events, such as those pertaining to sockets and interfaces.
lame-delegation (13)	Lame delegation events; although enabled by default, disabling this flag could prevent the log from getting filled with frequent lame delegation encounters.
root-query (14)	Queries and responses from root servers.
ddns-refreshes (15)	DNS update refreshes for Windows 2000 clients (disabled by default).
ddns-refreshes-details (16)	RRs refreshed during DNS updates for Windows 2000 clients (disabled by default).
ddns-details (17)	RRs added or deleted due to DNS updates.
tsig (18)	Logs events associated with Transaction Signature (TSIG) DNS updates (see the <a href="#">“Transaction Security”</a> section on page 27-6).
tsig-details (19)	More detailed logging of TSIG DNS updates (disabled by default).
activity-summary (20)	Summary of activities in the server. You can adjust the interval at which these summaries are taken using the <i>activity-summary-interval</i> attribute, which defaults to five-minute intervals (you can adjust this interval using <b>dns set-activity-summary-interval</b> ).
query-errors (21)	Logs errors encountered while processing DNS queries.
config-details (22)	Generates detailed information during server configuration by displaying all configured and assumed server attributes (disabled by default).
ha-details (30)	Generates detailed logging of High-Availability (HA) DNS information.

- Using the **nslookup** utility to test and confirm the DNS configuration—This utility is a simple resolver that sends queries to Internet nameservers. To obtain help for the **nslookup** utility, enter **help** at the prompt after you invoke the command. Use only fully qualified names with a trailing dot to ensure that the lookup is the intended one. An **nslookup** begins with a reverse query for the nameserver itself, which may fail if the server cannot resolve this due to its configuration. Use the **server** command, or specify the server on the command line, to ensure that you query the proper server. Use the **-debug**, or better yet, the **-d2**, flag to dump the responses and (with **-d2**) the queries being sent.

