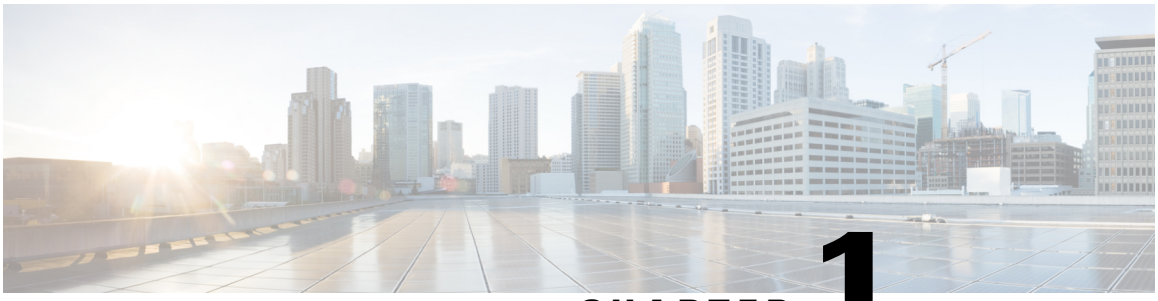
IP Addressing: DNS Configuration Guide, Cisco IOS Release 15S

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS



CHAPTER 1

Configuring DNS

The Domain Name System (DNS) is a distributed database in which you can map hostnames to IP addresses through the DNS protocol from a DNS server. Each unique IP address can have an associated hostname. The Cisco IOS software maintains a cache of hostname-to-address mappings for use by the **connect**, **telnet**, and **ping** EXEC commands, and related Telnet support operations. This cache speeds the process of converting names to addresses.

Note You can specify IPv4 and IPv6 addresses while performing various tasks in this feature. The resource record type AAAA is used to map a domain name to an IPv6 address. The IP6.ARPA domain is defined to look up a record given an IPv6 address.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Configuring DNS

To use DNS, you must have a DNS name server on your network.

Information About DNS

DNS Overview

If your network devices require connectivity with devices in networks for which you do not control name assignment, you can assign device names that uniquely identify your devices within the entire internetwork. The global naming scheme of the Internet, the DNS, accomplishes this task. This service is enabled by default. The following sections summarize DNS concepts and function.

Hostnames for Network Devices

Each unique IP address can have an associated hostname. DNS uses a hierarchical scheme for establishing hostnames for network nodes. This allows local control of the segments of the network through a client-server scheme. The DNS system can locate a network device by translating the hostname of the device into its associated IP address.

Domains Names for Groups of Networks

IP defines a naming scheme that allows a device to be identified by its location in the IP. This is a hierarchical naming scheme that provides for *domains*. On the Internet, a domain is a portion of the naming hierarchy tree that refers to general groupings of networks based on organization type or geography. Domain names are pieced together with periods (.) as the delimiting characters. For example, Cisco is a commercial organization that the IP identifies by a *com* domain name, so its domain name is *cisco.com*. A specific device in this domain, the File Transfer Protocol (FTP) system, for example, is identified as *ftp.cisco.com*.

Name Servers

To keep track of domain names, IP has defined the concept of a *name server*. Name servers are programs that have complete information about their namespace portion of the domain tree and may also contain pointers to other name servers that can be used to lead to information from any other part of the domain tree. Name servers know the parts of the domain tree for which they have complete information. A name server may also store information about other parts of the domain tree. Before domain names can be mapped to IP addresses, you must first identify the hostnames, then specify a name server, and enable the DNS service.

Cache

To speed the process of converting names to addresses, the name server maintains a database, called a *cache*, of hostname-to-address mappings for use by the **connect**, **telnet**, and **ping** EXEC commands, and related Telnet support operations. The cache stores the results from previous responses. Upon receiving a client-issued DNS query, the name server will check this local storage to see if the answer is available locally.

Name Resolvers

Name resolvers are programs that extract information from name servers in response to client requests. Resolvers must be able to access at least one name server. The resolver either uses that name server's information to answer a query directly or pursues the query using referrals to other names servers. A resolver will typically be a system routine that is directly accessible to user programs. Therefore, no protocol is necessary between the resolver and the user program.

Zones

The domain namespace is divided into areas called zones that are points of delegation in the DNS tree. A zone contains all domains from a certain point downward, except those for which other zones are authoritative.

Authoritative Name Servers

A name server is said to be an authority for the parts of the domain tree for which it has complete information. A zone usually has an authoritative name server, often more than one. An authoritative name server has been configured with host table information or has acquired host table information though a zone transfer (the action that occurs when a secondary DNS server starts up and updates itself from the primary server).

DNS Operation

An organization can have many name servers, but Internet clients can query only those that the root name servers know. The other name servers answer internal queries only.

A name server handles client-issued queries to the DNS server for locally defined hosts within a particular zone as follows:

- An authoritative name server responds to DNS user queries for a domain name that is under its zone of authority by using the permanent and cached entries in its own host table. If the query is for a domain name that is under its zone of authority but for which it does not have any configuration information, the authoritative name server simply replies that no such information exists.
- A name server that is not configured as the authoritative name server responds to DNS user queries by using information that it has cached from previously received query responses. If no device is configured as the authoritative name server for a zone, queries to the DNS server for locally defined hosts will receive nonauthoritative responses.

Name servers answer DNS queries (forward incoming DNS queries or resolve internally generated DNS queries) according to the forwarding and lookup parameters configured for the specific domain.

When DNS queries are forwarded to name servers for resolution, some memory space is held for the corresponding DNS query until an appropriate response is received or until there is timeout. To avoid the free I/O memory from getting exhausted when handling queries at high rate, configure the maximum size for the queue.

How to Configure DNS

Mapping Hostnames to IP Addresses

Perform this task to map hostnames to IP addresses.

A name server is used to keep track of information associated with domain names. A name server can maintain a database of hostname-to-address mappings. Each name can map to one or more IP addresses. In order to use this service to map domain names to IP addresses, you must specify a name server.

The name lookup system can be statically configured using the commands described in this task. Some other functions in Cisco IOS software, such as DHCP, can dynamically modify the state of the name lookup system. Use the **show hosts** command to display the cached hostnames and the DNS configuration.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip host** *name* [*tcp-port-number*] *address1* [*address2 ... address8*]
4. Do one of the following:
   - **ip domain name** *name*
   - **ip domain list** *name*
5. **ip name-server** *server-address1* [*server-address2 ... server-address6*]
6. **ip domain lookup** [**source-interface** *interface-type interface-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip host** *name* [*tcp-port-number*] *address1* [*address2 ... address8*]	Defines a static hostname-to-address mapping in the hostname cache. • The host IP address can be an IPv4 or IPv6 address.

	Command or Action	Purpose
	Example: `Device(config)# ip host cisco-rtp 192.168.0.148`	• Typically, it is easier to refer to network devices by symbolic names rather than numerical addresses (services such as Telnet can use hostnames or addresses). Hostnames and IP addresses can be associated with one another through static or dynamic means. • Manually assigning hostnames to addresses is useful when dynamic mapping is not available.

Step 4	Do one of the following:
	• **ip domain name** *name*
	• **ip domain list** *name*
	Example:
	`Device(config)# ip domain name cisco.com`
	Example:
	Example:
	`Device(config)# ip domain list cisco1.com`
		(Optional) Defines a default domain name that the Cisco IOS software will use to complete unqualified hostnames.
		or
		(Optional) Defines a list of default domain names to complete unqualified hostnames.
		• You can specify a default domain name that the Cisco IOS software will use to complete domain name requests. You can specify either a single domain name or a list of domain names. Any hostname that does not contain a complete domain name will have the default domain name you specify appended to it before the name is looked up.
		Note If there is no domain list, the domain name that you specified with the **ip domain name** global configuration command is used. If there is a domain list, the default domain name is not used. The **ip domain list** command is similar to the **ip domain name** command, except that with the **ip domain list** command you can define a list of domains, each to be tried in turn until the system finds a match.

Step 5	**ip name-server** *server-address1* [*server-address2 ... server-address6*] **Example:** `Device(config)# ip name-server 172.16.1.111 172.16.1.2`	Specifies one or more hosts (up to six) that can function as a name server to supply name information for DNS.
Step 6	**ip domain lookup** [**source-interface** *interface-type interface-number*] **Example:** `Device(config)# ip domain lookup`	(Optional) Enables DNS-based address translation. • DNS is enabled by default. Use this command if DNS has been disabled.

Customizing DNS

Perform this task to customize your DNS configuration.

In a multiple server configuration without the DNS round-robin functionality, many programs will use the first host server/IP address for the whole time to live (TTL) of the cache and use the second and third host servers/IP addresses only in the event of host failure. This behavior presents a problem when a high volume of users all arrive at the first host during the TTL time. For example, the network access server (NAS) sends out a DNS query. The DNS servers reply with a list of the configured IP addresses to the NAS. The NAS then caches these IP addresses for a given time (for example, five minutes). All users that dial in during the five minute TTL time will land on one host, the first IP address in the list.

In a multiple server configuration with the DNS round-robin functionality, the DNS server returns the IP address of all hosts to rotate between the cache of hostnames. During the TTL of the cache, users are distributed among the hosts. This functionality distributes calls across the configured hosts and reduces the number of DNS queries.

In a scheduling algorithm, processes are activated in a fixed cyclic order. Processes that are waiting for other events, like termination of a child process or an input or output operation, cannot proceed and hence they return control to the scheduler. If the TTL of the process times out just before the event (for which it was waiting) occurs, then the event will not be handled until all the other processes are activated.

Note The DNS round-robin functionality is applicable only for the DNS lookups on a device and is not applicable to another client pointing to the device.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip domain timeout** *seconds*
4. **ip domain retry** *number*
5. **ip domain round-robin**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip domain timeout** *seconds* **Example:** `Device(config)# ip domain timeout 17`	(Optional) Specifies the amount of time to wait for a response to a DNS query.

	Command or Action	Purpose
		• If the **ip domain timeout** command is not configured, the Cisco IOS software will wait 3 seconds for a response to a DNS query.
Step 4	**ip domain retry** *number* **Example:** `Device(config)# ip domain retry 10`	(Optional) Specifies the number of times to retry sending DNS queries. • If the **ip domain retry** command is not configured, the Cisco IOS software will retry DNS queries twice.
Step 5	**ip domain round-robin** **Example:** `Device(config)# ip domain round-robin`	(Optional) Enables round-robin functionality on DNS servers.

Configuring DNS Spoofing

Perform this task to configure DNS spoofing.

DNS spoofing is designed to allow a device to act as a proxy DNS server and "spoof" replies to any DNS queries using either the configured IP address in the **ip dns spoofing** *ip-address* command or the IP address of the incoming interface for the query. This feature is useful for devices where the interface toward the Internet service provider (ISP) is not up. Once the interface to the ISP is up, the device forwards DNS queries to the real DNS servers.

This feature turns on DNS spoofing and is functional if any of the following conditions are true:

- The **no ip domain lookup** command is configured.
- IP name server addresses are not configured.
- There are no valid interfaces or routes for sending to the configured name server addresses.

If these conditions are removed, DNS spoofing will not occur.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip dns server**
4. **ip dns spoofing** [*ip-address*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip dns server** **Example:** `Device(config)# ip dns server`	Activates the DNS server on the device.
Step 4	**ip dns spoofing** [*ip-address*] **Example:** `Device(config)# ip dns spoofing 192.168.15.1`	Configures DNS spoofing. • The IP address used for DNS spoofing can be an IPv4 or IPv6 address. • The device will respond to the DNS query with the configured *ip-address* when queried for any hostname other than its own. • The device will respond to the DNS query with the IP address of the incoming interface when queried for its own hostname.

Configuring the Device as a DNS Server

Perform this task to configure the device as a DNS server.

A Cisco IOS device can provide service to DNS clients, acting as both a caching name server and as an authoritative name server for its own local host table.

When configured as a caching name server, the device relays DNS requests to other name servers that resolve network names into network addresses. The caching name server caches information learned from other name servers so that it can answer requests quickly, without having to query other servers for each transaction.

When configured as an authoritative name server for its own local host table, the device listens on port 53 for DNS queries and then answers DNS queries using the permanent and cached entries in its own host table.

An authoritative name server usually issues zone transfers or responds to zone transfer requests from other authoritative name servers for the same zone. However, the Cisco IOS DNS server does not perform zone transfers.

When it receives a DNS query, an authoritative name server handles the query as follows:

- If the query is for a domain name that is not under its zone of authority, the authoritative name server determines whether to forward the query to specific back-end name servers based on whether IP DNS-based hostname-to-address translation has been enabled via the **ip domain lookup** command.
- If the query is for a domain name that is under its zone of authority and for which it has configuration information, the authoritative name server answers the query using the permanent and cached entries in its own host table.
- If the query is for a domain name that is under its zone of authority but for which it does not have any configuration information, the authoritative name server does not forward the query elsewhere for a response; instead the authoritative name server simply replies that no such information exists.

Note Unless Distributed Director is enabled, the TTL on locally defined resource records will always be ten seconds, regardless of any authority record parameters that may have been specified for the DNS name server by the use of the **ip dns primary** command.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip dns server**
4. **ip name-server** *server-address1* [*server-address2... server-address6*]
5. **ip dns server queue limit** {**forwarder** *queue-size-limit* | **director** *queue-size-limit*}
6. **ip host** [**vrf** *vrf-name*] [**view** *view-name*] *hostname* {*address1* [*address2 ... address8*] | **additional** *address9* [*address10 ... addressn*]}
7. **ip dns primary** *domain-name* **soa** *primary-server-name* *mailbox-name* [*refresh-interval* [*retry-interval* [*expire-ttl* [*minimum-ttl*]]]]
8. **ip host** *domain-name* **ns** *server-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**ip dns server** **Example:** `Device(config)# ip dns server`	Enables the DNS server.
Step 4	**ip name-server** *server-address1* [*server-address2... server-address6*] **Example:** `Device(config)# ip name-server 192.168.2.120 192.168.2.121`	(Optional) Configures other DNS servers: • Cisco IOS resolver name servers • DNS server forwarders **Note** If the Cisco IOS name server is being configured to respond only to domain names for which it is authoritative, there is no need to configure other DNS servers.
Step 5	**ip dns server queue limit** {**forwarder** *queue-size-limit* | **director** *queue-size-limit*} **Example:** `Device(config)# ip dns server queue limit forwarder 10`	(Optional) Configures a limit to the size of the queues used by the DNS server processes. • The **director** keyword was removed in Cisco IOS Release 12.4(24)T.
Step 6	**ip host** [**vrf** *vrf-name*] [**view** *view-name*] *hostname* {*address1* [*address2 ... address8*] | **additional** *address9* [*address10 ... addressn*]} **Example:** `Device(config)# ip host user1.example.com 192.168.201.5 192.168.201.6`	(Optional) Configures local hosts.
Step 7	**ip dns primary** *domain-name* **soa** *primary-server-name mailbox-name* [*refresh-interval* [*retry-interval* [*expire-ttl* [*minimum-ttl*]]]] **Example:** `Device(config)# ip dns primary example.com soa ns1.example.com mb1.example.com`	Configures the device as the primary DNS name server for a domain (zone) and as the start of authority (SOA) record source (which designates the start of a zone). **Note** Unless Distributed Director is enabled, the TTL on locally defined resource records will always be ten seconds.
Step 8	**ip host** *domain-name* **ns** *server-name* **Example:** `Device(config)# ip host example.com ns ns1.example.com`	(Optional) Configures the device to create an name server (NS) resource record to be returned when the DNS server is queried for the associated domain. • This configuration is needed only if the zone for which the system is authoritative will also be served by other name servers.

Examples

This section provides examples of debugging output that is logged when a device is configured as an authoritative name server for its own local host table and the **debug domain** command is in effect:

Note For DNS-based X.25 routing, the **debug x25 events** command supports functionality to describe the events that occur while the X.25 address is being resolved to an IP address using a DNS server. The **debug domain** command can be used along with **debug x25 events** to observe the whole DNS-based X.25 routing data flow.

Debugging Output for Relaying a DNS Query to Another Name Server Example

The following is sample output from the **debug domain** command that corresponds to relaying a DNS query to another name server when the device is configured as an authoritative name server for its own local host table:

```
Apr  4 22:18:32.183: DNS: Incoming UDP query (id#18713)
Apr  4 22:18:32.183: DNS: Type 1 DNS query (id#18713) for host 'ns1.example.com' from
192.0.2.120(1283)
Apr  4 22:18:32.183: DNS: Re-sending DNS query (type 1, id#18713) to 192.0.2.121
Apr  4 22:18:32.211: DNS: Incoming UDP query (id#18713)
Apr  4 22:18:32.211: DNS: Type 1 response (id#18713) for host <ns1.example.com> from
192.0.2.121(53)
Apr  4 22:18:32.215: DOM: dom2cache: hostname is ns1.example.com, RR type=1, class=1,
ttl=86400, n=4
Apr  4 22:18:32.215: DNS: Forwarding back A response - no director required
Apr  4 22:18:32.215: DNS: Finished processing query (id#18713) in 0.032 secs
Apr  4 22:18:32.215: DNS: Forwarding back reply to 192.0.2.120/1283
```

Debugging Output for Servicing a DNS Query from the Local Host Table Example

The following is sample output from the **debug domain** command that corresponds to servicing a DNS query from the local host table when the device is configured as an authoritative name server for its own local host table:

```
Apr  4 22:16:35.279: DNS: Incoming UDP query (id#8409)
Apr  4 22:16:35.279: DNS: Type 1 DNS query (id#8409) for host 'ns1.example.com' from
192.0.2.120(1279)
Apr  4 22:16:35.279: DNS: Finished processing query (id#8409) in 0.000 secs
```

Disabling DNS Queries for ISO CLNS Addresses

Perform this task to disable DNS queries for International Organization for Standardization (ISO) Connectionless Network Service (CLNS) addresses.

If your device has both IP and ISO CLNS enabled and you want to use ISO CLNS network service access point (NSAP) addresses, you can use the DNS to query these addresses, as documented in RFC 1348. This feature is enabled by default.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **no ip domain lookup nsap**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no ip domain lookup nsap** **Example:** `Device(config)# no ip domain lookup nsap`	Disables DNS queries for ISO CLNS addresses.

Verifying DNS

Perform this task to verify your DNS configuration.

1 **enable**

2 **ping** *hosts*

3 **show hosts**

SUMMARY STEPS

1. **enable**
2. **ping** *hosts*
3. **show hosts**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**ping** *hosts* **Example:** `Device# ping cisco-rtp`	Diagnoses basic network connectivity. • After the DNS configuration is set, you can verify the DNS server by using a hostname to ping or telnet to a device.
Step 3	**show hosts** **Example:** `Device# show hosts`	Displays the default domain name, the style of name lookup service, a list of name server hosts, and the cached list of hostnames and addresses. • After a name is resolved using DNS, use the **show hosts** command to view the cached hostnames and the DNS configuration.

Configuration Examples for DNS

Example: IP Addresses

The following example establishes a domain list with several alternate domain names:

```
ip domain list example.com
ip domain list example1.edu
ip domain list example2.edu
```

Example: Mapping Hostnames to IP Addresses

The following example configures the hostname-to-address mapping process. IP DNS-based translation is specified, the addresses of the name servers are specified, and the default domain name is given.

```
! IP DNS-based hostname-to-address translation is enabled
ip domain lookup
! Specifies hosts 192.168.1.111 and 192.168.1.2 as name servers
ip name-server 192.168.1.111 192.168.1.2
! Defines cisco.com as the default domain name the device uses to complete
! Set the name for unqualified hostnames
ip domain name cisco.com
```

Example: Customizing DNS

The following example allows a Telnet to company.example.com to connect to each of the three IP addresses specified in the following order: the first time the hostname is referenced, it would connect to 10.0.0.1; the second time the hostname is referenced, it would connect to 10.1.0.1; and the third time the hostname is referenced, it would connect to 10.2.0.1. In each case, the other two addresses would also be tried if the first one failed; this is the normal operation of the Telnet command.

```
Device(config)# ip host company.example.com 10.0.0.1 10.1.0.1 10.2.0.1
Device(config)# ip domain round-robin
```

Example: Configuring DNS Spoofing

In the following example, the device is configured to spoof replies to any DNS queries:

```
ip dns server
ip dns spoofing
no ip domain lookup
interface e3/1
 ip address 10.1.1.1 255.255.255.0
```

Additional References

Related Documents

Related Topic	Document Title
DNS commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS IP Addressing Services Command Reference*

Standards

Standards	Title
No new or modified standards are supported by this functionality.	--

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
RFC 1348	*DNS NSAP RRs*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

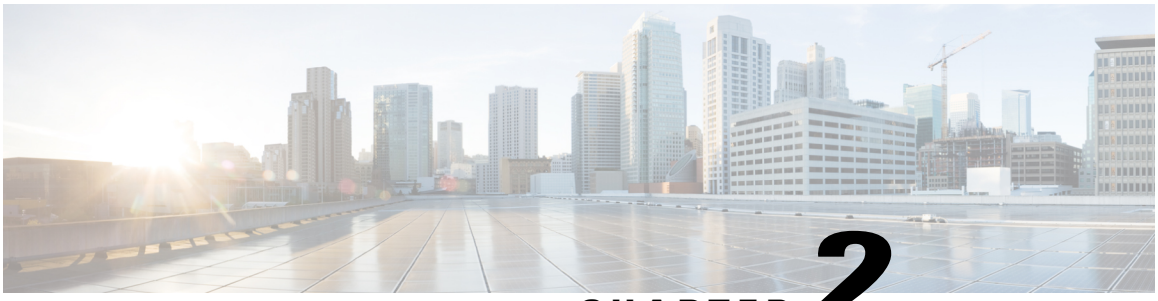
Feature Information for DNS

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 1: Feature Information for DNS

Feature Name	Releases	Feature Information
DNS Spoofing	12.3(2)T 15.4(1)T	This feature is designed to allow a device to act as a proxy DNS server and "spoof" replies to any DNS queries using either the configured IP address in the **ip dns spoofing** *ip-address* command or the IP address of the incoming interface for the query. The following command was introduced by this feature: **ip dns spoofing**.



CHAPTER 2

Service Discovery Gateway

The Service Discovery Gateway feature enables multicast Domain Name System (mDNS) to operate across Layer 3 (L3) boundaries. An mDNS gateway will be able to provide transport for service discovery across L3 boundaries by filtering, caching and extending services from one subnet to another. Prior to implementation of this feature, mDNS was limited in scope to within a subnet due to the use of link-local scoped multicast addresses. This feature enhances Bring Your Own Device (BYOD).

Caution Extension of services should be done with proper care. Generally, only specific services should be extended. Service names should be unique in the network to avoid duplicate name conflicts.

See Feature Information for Service Discovery Gateway section to check feature availability for your platform release version.

Information About Service Discovery Gateway

Service Announcement Redistribution and Service Extension

Redistribution of announcements is the actual forwarding of announcements and query responses while service extension is the capability of proxying services between subnets. The actual replication of the service announcement can help to speed up the visibility of newly announced services and also a service's withdrawal if a service or device is turned off.

Note Extension of services such as printers or Apple TV works fine without actual replication of service announcements. The Service Discovery Gateway will cache announcements, queries and their responses in the cache. If another device queries for a service, the Service Discovery Gateway will be able to provide an answer from its cache.

Enable the **redistribution mdns-sd** command only on a per-interface basis, and only if it is actually required. You must ensure that there are no loops in the network topology corresponding to the interface for which service announcement redistribution is being enabled. A loop can lead to a broadcast storm.

Redistribution of service announcement information cannot be done globally. You can enable redistribution of service information only at the interface level.

Extending Services Across Subnets—An Overview

You need to enable a multicast Domain Name System (mDNS) gateway to extend services across subnet boundaries. You can enable an mDNS gateway for a device or for an interface. You must enable routing of services for the device before enabling it at the interface level. After the mDNS gateway is enabled on a device or interface, you can extend services across subnet boundaries.

To extend services across subnets, you must do the following:

1. Set Filter Options to Extend Services Across Subnets—You can allow services such as printer services to be accessed across subnets. If printer x is available on interface 1, users on interface 2 can use printer x without configuring the printer on their local systems.
2. Extend Services Across Subnets—The filter created in Step 1 should be applied on the interfaces 1 and 2. Only then can users on other interfaces access the printer service.

For the sample scenario where a printer service is accessible by clients on other interfaces, you must apply these filters:

- On the interface where the printer service is available (IN filter) —You want to allow the printer service *into* the mDNS cache, so that it can be accessed by users on other subnets.
- On the interface where the printer service is available (OUT filter)—Since clients on other interfaces will access the service (printer x, for example), you should allow queries coming from the device (OUT filter, from the device's point of view).
- On each interface where clients reside (IN filter)—For clients on other interfaces (subnets) wanting to access the printer service, you must allow queries from users into the mDNS cache (IN filter).

Remember Applying the IN filter means that you are allowing the printer service into the device mDNS cache, and other interfaces can access it. Applying the OUT filter means that you are allowing the queries out of the cache so that queries from clients on other interfaces can reach the printer interface. On other client-facing interfaces, the IN filter is applied to allow queries in.

Note

- Filters can be applied at the global level and at the interface level. Filters applied at the interface level takes precedence over the filters applied at the global level.
- The term 'service discovery information' refers to services (printer services, etc), queries (queries for printer services, etc, from one interface to the other), announcements (printer service is removed, etc), and service-instances (a specific service—printer x, Apple TV 3, etc) that you want to extend across subnets.

Set Filter Options to Extend Services Across Subnets

You can set filter options to allow services such as printer services into or out of a device or interface. You can also permit or prohibit queries, announcements, services learnt from an interface, specific service–instances, and locations. Use the **service-list mdns-sd** command to create a service-list and set filter options.

You need to create a service-list and use filter options within it. While creating a service-list, use one of the following options:

- The **permit** option permits specific services, announcements and service-instances across subnets.
- The **deny** option restricts services, announcements and service-instances from being transported across subnets.
- The **query** option is provided to browse services. For example, if you want to browse printer services periodically, then you can create a service-list with the **query** option, and add the printer service to the query. When you set a period for the query, the service entries are refreshed in the cache memory.

You must mention a sequence number when using the **permit** or **deny** option. The filtering is done sequentially, in the ascending order. The same service-list can be associated with multiple sequence numbers. Within a sequence, match statements (commands) must be used to specify what needs to be filtered. Generally, match statements are used to filter queries (for example, queries from clients to find printer and fax services), announcements (new service is added, and so on), specific service–instances, types of service such as printer services (so that the service is allowed into the cache for use), services available for a specific interface (printers and Apple TVs associated with a VLAN), and locations.

Note A service-list by itself does not contain any services. You must specify a service type in the match statement when setting filter options to allow or prohibit services. (For example, '_ipp._tcp' is the service type for an IPP printing service running over TCP).

Sample scenario - Consider a device is in a client segment. The goal is to allow the following on the device:

- All queries from clients to the device.
- Printer services to clients on other subnets.

The following example explains how to achieve the goal:

```
!
service-list mdns-sd mixed permit 10
  match message-type query
```

```
!
service-list mdns-sd mixed permit 20
  match message-type announcement
  match service-type _ipps._tcp.local
!
```

In the above example, a service-list called 'mixed' is created and the **permit** option is used twice—to filter queries and to filter printer services and announcements. The filtering is done in the sequence given below:

- Sequence 10 - A match statement is used to filter queries.
- Sequence 20 - Match statements are used to filter announcements and printer services.

The match statement in Sequence 10 sets a filter for queries on the device, but does not specify that queries be allowed *into* the device. To allow queries from clients, the filter needs to be applied on the interface in the IN direction. The example is displayed in the Extend Services Across Subnets section.

Similarly, the match statements in Sequence 20 sets a filter for announcements and printer services on the device, but does not specify that they be allowed *into* the device. To allow announcements and printer services into the device, the filter needs to be applied on the required interfaces in the IN direction. The example is displayed in the Extend Services Across Subnets section.

If neither the **permit** option nor the **deny** option is used, the default action is to disallow services from being transported to other subnets.

Browsing services periodically—Service-lists of the type **query** can be used to browse services. Such queries are called active queries. Active queries periodically send out requests for the services specified within the query on all interfaces. As services have a specific Time to Live (TTL) duration, active queries can help to keep services fresh in the cache memory.

In the following example, a service-list named 'active-query' is created and the service-list is of the type **query**. Services such as printer services are specified within the query, and these are the services that we want to extend. Typically, these services would match the services that have been configured as 'permitted' services in the IN filter.

```
!
service-list mdns-sd active-query query
  service-type _universal._sub._ipp._tcp
  service-type _ipp._tcp.local
  service-type _ipps._tcp.local
  service-type _raop._tcp.local
!
```

The purpose of an active query and a query associated with a match statement is different. When you enable an active query, services are browsed periodically. A query is used in a match statement to permit or prohibit queries (not active queries) on the interface.

Note

- Service-list creation can only be used globally and cannot be used at the interface level.
- You can create a new service-instance of a specific service-type using the **service-instance mdns-sd** command.
- A service end–point (such as a printer, fax, and so on) sends unsolicited announcements when a service starts up. After that, it sends unsolicited announcements whenever a network change event occurs (such as, an interface coming up or going down, and so on). The device always responds to queries.

Remember Filtering only sets filter options and specifies that certain services need to be filtered. You must *apply* the filters on an interface for the services, queries, or announcements to actually be permitted or prohibited on the interface. To know about applying filters and the other available service discovery configuration options, refer the Extend Services Across Subnets section.

Extend Services Across Subnets

You must have set filter options for the device before extending services across subnets. If you have set filter options for specific services and other service discovery information to be allowed, prohibited or queried periodically, you can apply the filters for an interface.

Before applying filters, note the following:

- You must enable multicast Domain Name System (mDNS) on a device to apply filter options. You can enable mDNS using the command **service-routing mdns-sd**
- Since you might want to allow services into the device or prohibit services from being learnt on an interface, you must apply the filter in the needed direction. The options **IN** and **OUT** perform the desired actions on the interface.
- Typically, a service-policy is applied on an interface. Global service-policies are optional and affect all L3 interfaces.

Sample scenario - A device is in a client segment and the goal is to allow the following between the device interfaces:

- All queries from clients to the device.
- Printer services.

A note about filter options - Filter options have been set for the above scenario by creating a service-list called 'mixed' and adding filter options to it. (see Set Filter Options to Extend Services Across Subnets for more details). The following example explains how to apply the filters:

```
!
interface Ethernet0/0
 description *** (wireless) Clients here plus some printers
 ip address 172.16.33.7 255.255.255.0
 service-routing mdns-sd
  service-policy mixed IN
!
interface Ethernet0/3
 description *** (wireless) Clients here plus some printers
 ip address 172.16.57.1 255.255.255.0
 service-routing mdns-sd
  service-policy mixed IN
!
```

In the above example, service-routing is enabled on the interface and the filter options in the service-policy 'mixed' are applied in the **IN** direction. In other words, all queries and printer services will be allowed into the device, from the interfaces Ethernet 0/0 and Ethernet 0/3.

Sample scenario for browsing specific services - A service-list of the type **query** (called active query) has been created. It contains services that we want to browse periodically, such as printer services (see Set Filter

Options to Extend Services Across Subnets for more details about creating an active query). To enable browsing of the services in the query, you must apply the active query for the device.

```
!
service-routing mdns-sd
 service-policy-query active-query 900
!
```

In the above example, the period is set to 900 seconds. The services within the active query are queried on all interfaces of the device after an interval of 900 seconds.

Note

- You can enable browsing of services for specific interfaces. If browsing of services is enabled globally, you can disable browsing of services on specific interfaces.
- Services are browsed specific to a device or interface by the mDNS process. So, the IN or OUT option is not relevant for browsing of services.

You can use the following options after enabling mDNS on a device or interface.

Purpose	**Use this Command** **Note** The complete syntax is provided in the corresponding task.	**Global and Interface Configuration Options**
For a service-list, apply a filter to allow or prohibit services.	**service-policy**	Global and interface levels.
Set some part of the system memory for cache.	**cache-memory-max**	Global level.
Configure an active query and the query period so that specified services are queried periodically.	**service-policy-query**	
Designate a specific device or interface in a domain for routing mDNS announcement and query information.	**designated-gateway**	Global and interface levels.
Access services in the proximity of the device. **Note** Service policy proximity filtering functionality is only available on wireless devices and their interfaces.	**service-policy-proximity**	Global and interface levels.
Configure service-type enumeration period for the device.	**service-type-enumeration period**	Global level.

Specify an alternate source interface for outgoing mDNS packets on a device.	**source-interface**	Global level.
Configure the maximum rate limit of incoming mDNS packets for a device.	**rate-limit**	Global level.
Speed up visibility of newly announced services and withdrawal of services when a service or device is turned off.	**redistribute**	Interface level.

How to Configure Service Discovery Gateway

Setting Filter Options for Service Discovery

Before You Begin

Ensure that you permit a query or announcement when you set filter options. If you do not use a **permit** option and only use **deny** options, you will not be able to apply the filter.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **service-list mdns-sd** *service-list-name* {**deny** *sequence-number* | **permit** *sequence-number* | **query**}
4. **match message-type** {**announcement** | **any** | **query**}
5. **match service-instance** {*instance-name* | **any** | **query**}
6. **match service-type** *mDNS-service-type-string*
7. **match location civic** *civic-location-name*
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 3	**service-list mdns-sd** *service-list-name* {**deny** *sequence-number* | **permit** *sequence-number* | **query**}
	
	Example:
	`Device(config)# service-list mdns-sd sl1 permit 3`
	
	`Or`
	
	`Device(config)# service-list mdns-sd sl4 query`
		Enters mdns service discovery service-list mode.
		• Creates a service-list and applies a filter on the service-list according to the **permit** or **deny** option applied to the sequence number.
		
		Or
		
		• Creates a service-list and associates a query for the service-list name if the **query** option is used.
		
		Remember When you set filter options, ensure that you permit a query or announcement for a service-list. If you do not use a **permit** option and only use **deny** options, you will not be able to apply the filter.

Step 4	**match message-type** {**announcement** | **any** | **query**} **Example:** `Device(config-mdns-sd-sl)# match message-type announcement`	Configures parameters for a service-list based on a service announcement or query. **Note** You cannot use the **match** command if you have used the **query** option. The **match** command can be used only for the **permit** or **deny** option.
Step 5	**match service-instance** {*instance-name* | **any** | **query**} **Example:** `Device(config-mdns-sd-sl)# match service-instance printer-3`	Configures parameters for a service-list based on a service-instance or query.
Step 6	**match service-type** *mDNS-service-type-string* **Example:** `Device(config-mdns-sd-sl)# match service-type _ipp._tcp.local`	Configures parameters for a service-list based on a service-type.
Step 7	**match location civic** *civic-location-name* **Example:** `Device(config-mdns-sd-sl)# match location civic location3`	Configures parameters for a service-list based on a civic location.

	Command or Action	Purpose
Step 8	**exit** **Example:** `Device(config-mdns-sd-sl)# exit`	Exits mdns service discovery service-list mode, and returns to global configuration mode.

What to Do Next

Apply filters on an interface for the services, queries, or announcements to actually be permitted or prohibited on the interface.

Applying Service Discovery Filters and Configuring Service Discovery Parameters

After enabling multicast Domain Name System (mDNS) gateway for a device, you can apply filters (IN-bound filtering or OUT-bound filtering) and active queries by using **service-policy** and **service-policy-query** commands, respectively.

Note Steps 5 to 11 are mDNS Service Discovery configuration options. The steps are optional and not meant to be used in any specific order.

Before You Begin

You must set filter options for the device before applying filters.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **service-routing mdns-sd**
4. **service-policy** *service-policy-name* {**IN** | **OUT**}
5. **cache-memory-max** *cache-config-percentage*
6. **service-policy-query** *service-list-name query-period*
7. **designated-gateway enable** [**ttl** *duration*]
8. **service-policy-proximity** *service-list-name* [**limit** *number-of-services*]
9. **service-type-enumeration period** *period-value*
10. **source-interface** *type number*
11. **rate-limit in** *maximum-rate-limit*
12. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**service-routing mdns-sd** **Example:** `Device(config)# service-routing mdns-sd`	Enables mDNS gateway functionality for a device and enters multicast DNS configuration (config-mdns) mode.
Step 4	**service-policy** *service-policy-name* {**IN** | **OUT**} **Example:** `Device(config-mdns)# service-policy sl1 IN`	For a service-list, applies a filter on incoming service discovery information (IN-bound filtering) or outgoing service discovery information (OUT-bound filtering). **Note** Global service-policies are optional and effect all L3 interfaces. Typically, a service-policy is applied on an interface.
Step 5	**cache-memory-max** *cache-config-percentage* **Example:** `Device(config-mdns)# cache-memory-max 20`	Sets some part of the system memory (in percentage) for cache. **Note** By default, 10% of the system memory is set aside for cache. You can override the default value by using this command.
Step 6	**service-policy-query** *service-list-name query-period* **Example:** `Device(config-mdns)# service-policy-query sl4 100`	Creates an active query and configures the service-list-query period.
Step 7	**designated-gateway enable** [**ttl** *duration*] **Example:** `Device(config-mdns)# designated-gateway enable`	Designates the device to route mDNS announcement and query information for the domain.
Step 8	**service-policy-proximity** *service-list-name* [**limit** *number-of-services*]	Configures service policy proximity filtering on the device. • Service policy proximity filtering is only available for wireless clients and is based on Radio Resource

	Command or Action	Purpose
	Example: `Device(config-mdns)# service-policy-proximity sl1 limit 10`	Management (RRM). Wired clients and services are not affected by the limit. • The default value for the maximum number of services that can be returned is 50.
Step 9	**service-type-enumeration period** *period-value* **Example:** `Device(config-mdns)# service-type-enumeration period 45`	Configures service-type enumeration period for the device.
Step 10	**source-interface** *type number* **Example:** `Device(config-mdns)# source-interface ethernet 0/1`	Specifies an alternate source interface for outgoing mDNS packets on a device.
Step 11	**rate-limit in** *maximum-rate-limit* **Example:** `Device(config-mdns)# rate-limit in 80`	Configures the maximum rate limit of incoming mDNS packets for a device.
Step 12	**exit** **Example:** `Device(config-mdns)# exit`	Exits multicast DNS configuration mode, and returns to global configuration mode.

Applying Service Discovery Filters for an Interface

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **service-routing mdns-sd**
5. **service-policy** *service-policy-name* {**IN** | **OUT**}
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface ethernet 0/1`	Enters Interface multicast DNS configuration mode, and enables interface configuration.
Step 4	**service-routing mdns-sd** **Example:** `Device(config-if)# service-routing mdns-sd`	Enables mDNS gateway functionality for an interface and enters multicast DNS configuration (config-mdns) mode.
Step 5	**service-policy** *service-policy-name* {**IN** | **OUT**} **Example:** `Device(config-if-mdns-sd)# service-policy sl1 IN`	For a service-list, applies a filter on incoming service discovery information (IN-bound filtering) or outgoing service discovery information (OUT-bound filtering). **Remember** When you set filter options, ensure that you permit a query or announcement for a service-list. If you have not permitted a service, query, or announcement while setting filter options, then you will see this warning when you apply the filter: **Warning**: Please enable explicit service-list rule with the permit action to allow queries and responses.
Step 6	**exit** **Example:** `Device(config-if-mdns-sd)# exit`	Exits Interface multicast DNS configuration mode, and returns to interface configuration mode.

Creating a Service Instance

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **service-instance mdns-sd service** *instance-name* **regtype** *service-type* **domain** *name*
4. {**ipv4addr** | **ipv6addr**} *IP-address*
5. **port** *number*
6. **target-hostname** *host-name*
7. **txt** *text-record-name*
8. **priority** *value*
9. **weight** *value*
10. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**service-instance mdns-sd service** *instance-name* **regtype** *service-type* **domain** *name* **Example:** `Device(config)# service-instance mdns-sd service printer-3 regtype _ipp._tcp.local domain tcp4`	Creates a service-instance of a specific service type and enters multicast Domain Name System (mDNS) service discovery service-instance (config-mdns-sd-si) mode. **Note** In this mode, you can configure various parameters for the service-instance. The subsequent steps show how to configure service-instance parameters.
Step 4	{**ipv4addr**	**ipv6addr**} *IP-address* **Example:** `Device(config-mdns-sd-si)# ipv4addr 209.165.200.230 255.255.255.0`	Specifies the IPv4 or IPv6 address of the port on which the service is available.

	Command or Action	Purpose
Step 5	**port** *number* **Example:** `Device(config-mdns-sd-si)# port 9100`	Specifies the port on which the service is available.
Step 6	**target-hostname** *host-name* **Example:** `Device(config-mdns-sd-si)# target-hostname fqdn-of-printer.example.com.`	Specifies the fully qualified domain name (FQDN) of the target host.
Step 7	**txt** *text-record-name* **Example:** `Device(config-mdns-sd-si)# txt _ipp._tcp.local=printer3`	Specifies the text record associated with the service instance. **Note** A TXT record is a type of DNS record that provides text information to sources outside your domain. Specify the text record in the format 'service-type=service-name'. To specify multiple records, use a semicolon (;) as a separator.
Step 8	**priority** *value* **Example:** `Device(config-mdns-sd-si)# priority 3`	(Optional) Specifies the priority value for the service-instance. The default priority value is zero.
Step 9	**weight** *value* **Example:** `Device(config-mdns-sd-si)# weight 20`	(Optional) Specifies the weight value for the service-instance. The default weight value is zero.
Step 10	**exit** **Example:** `Device(config-mdns-sd-si)# exit`	Exits multicast Domain Name System (mDNS) service discovery service-instance (config-mdns-sd-si) mode and enters global configuration mode.

Verifying and troubleshooting Service Discovery Gateway

Note The show and debug commands mentioned below are not in any specific order.

SUMMARY STEPS

1. **show mdns requests** [**detail** | [**type** *record-type*] [**name** *record-name*]]
2. **show mdns cache** [**interface** *type number* [**detail**] | [**name** *record-name*] [**type** *record-type*] [**detail**]]
3. **show mdns statistics** {**all** | **interface** *type number* | **service-list** *list-name* | [**cache** | **service-policy**] {**all** | **interface** *type number*} | **services orderby providers**}
4. **show mdns service-types** [**all** | **interface** *type number*]
5. **debug mdns** {**all** | **error** | **event** | **packet** | **verbose**}

DETAILED STEPS

Step 1 **show mdns requests** [**detail** | [**type** *record-type*] [**name** *record-name*]]

Example:

```
Device# show mdns requests detail

MDNS Outstanding Requests
=========================================================
Request name  :   _ipp._tcp.local
Request type  :   PTR
Request class :   IN
```

This command displays information for outstanding multicast Domain Name System (mDNS) requests, including record name and record type information.

Step 2 **show mdns cache** [**interface** *type number* [**detail**] | [**name** *record-name*] [**type** *record-type*] [**detail**]]

Example:

Note You can use the **detail** keyword for a specific interface, record or type. You cannot use it independently with the **show mdns cache** command.

```
Device# show mdns cache

mDNS CACHE
==========================================================================================
[<NAME>]                                                [<TYPE>][<CLASS>] [<TTL>/Remaining] [Accessed]
 [If-index] [<RR Record Data>]


_services._dns-sd._udp.local                              PTR     IN      4500/4496              0
     3       _ipp._tcp.local

_ipp._tcp.local                                           PTR     IN      4500/4496              1
     3       printer1._ipp._tcp.local

printer1._ipp._tcp.local                                  SRV     IN      120/116         1      3
     0       0       5678    much-WS.local

printer1._ipp._tcp.local                                  TXT     IN      4500/4496              1
     3       (1)''

music-WS.local                                          A       IN      120/116         1      3
  192.168.183.1
```

This command displays mDNS cache information.

Step 3 **show mdns statistics** {**all** | **interface** *type number* | **service-list** *list-name* | [**cache** | **service-policy**] {**all** | **interface** *type number*} | **services orderby providers**}

Example:

```
Device# show mdns statistics all

mDNS Statistics
 mDNS packets sent       : 0
 mDNS packets received   : 31
 mDNS packets dropped    : 8
 mDNS cache memory in use: 64264(bytes)
```

This command displays mDNS statistics.

Step 4 **show mdns service-types** [**all** | **interface** *type number*]

Example:

```
Device# show mdns service-types

mDNS SERVICES
================================
[<NAME>]          [<TTL>/Remaining] [If-name]
_ipp._tcp.local     4500/4496
```

This command displays mDNS statistics.

Step 5 **debug mdns** {**all** | **error** | **event** | **packet** | **verbose**}

Example:

```
Device# debug mdns all
```

This command enables all mDNS debugging flows.

Configuration Examples for Service Discovery Gateway

Example: Setting Filter Options for Service Discovery

The following example shows creation of a service-list sl1. The permit option is being applied on sequence number 3 and all services with message-type announcement are filtered and available for transport across various subnets associated with the device.

```
Device> enable
Device# configure terminal
Device(config)# service-list mdns-sd sl1 permit 3
Device(config-mdns-sd-sl)# match message-type announcement
Device(config-mdns-sd-sl)# exit
```

Example: Applying Service Discovery Filters and Configuring Service Discovery Parameters

```
Device> enable
Device# configure terminal
```

```
Device(config)# service-routing mdns-sd
Device(config-mdns)# service-policy serv-pol1 IN
Device(config-mdns)# cache-memory-max 20
Device(config-mdns)# service-policy-query sl-query1 100
Device(config-mdns)# designated-gateway enable
Device(config-mdns)# rate-limit in 80
Device(config-mdns)# exit
```

Example: Applying Service Discovery Filters for an Interface

Example: Setting Multiple Service Discovery Filter Options

The following example shows creation of filters using service-lists mixed, permit-most, permit-all, and deny-all. Then, the filters are applied at various interfaces, as required.

```
!
service-list mdns-sd mixed permit 10
 match message-type query
!
service-list mdns-sd mixed permit 20
 match message-type announcement
 match service-type _ipps._tcp.local
!
service-list mdns-sd mixed permit 30
 match message-type announcement
 match service-type _ipp._tcp.local
 match service-type _universal._sub._ipp._tcp
!
service-list mdns-sd mixed permit 40
 match message-type announcement
!
service-list mdns-sd mixed deny 50
!
service-list mdns-sd permit-most deny 10
 match service-type _sleep-proxy._udp.local
!
service-list mdns-sd permit-most permit 20
!
service-list mdns-sd permit-all permit 10
!
service-list mdns-sd deny-all permit 10
 match message-type query
!
service-list mdns-sd deny-all deny 20
!
service-list mdns-sd active-query query
 service-type _universal._sub._ipp._tcp.local
 service-type _ipp._tcp.local
 service-type _ipps._tcp.local
 service-type _raop._tcp.local
!
service-routing mdns-sd
 service-policy-query active-query 900
!
!
interface Ethernet0/0
 description *** (wireless) Clients here plus some printers or aTVs
 ip address 172.16.33.7 255.255.255.0
 service-routing mdns-sd
  service-policy mixed IN
  service-policy permit-all OUT
!
interface Ethernet0/1
 description *** AppleTVs, Print Servers here
```

```
 ip address 172.16.57.1 255.255.255.0
 service-routing mdns-sd
  service-policy permit-most IN
  service-policy permit-all OUT
!
interface Ethernet0/2
 description *** Clients only, we don't want to learn anything here
 ip address 172.16.58.1 255.255.255.0
 service-routing mdns-sd
  service-policy deny-all IN
  service-policy permit-all OUT
!
interface Ethernet0/3
 no ip address
 shutdown
!
```

In the above example, the service-lists are:

- permit-all - As the name suggests, this service-list permits all resource records, and should be used with care. This is typically applied in the OUT direction; allows the cache to respond to all requests regardless of query content or query type.
- permit-most - This allows anything in, except for sleep-proxy services. This is because extending sleep-proxy services causes an issue with devices that register with a sleep proxy across the Service Discovery Gateway. Due to split horizon, the real (sleeping) device won't be able to re-register its services when waking up again when its pointer (PTR) record is pointing to the sleep-proxy.
- deny-all - This prevents the cache from learning anything. Again incoming on a segment where only clients live. As a result, clients will be able to query for services from the cache (hence the permit 10 match query), but there is no need to learn anything from the clients.
- mixed - This is created to be used in client segments. In addition to clients (such as iPads, PCs, and so on), the occasional printer or a TV will also connect. The purpose here is to learn about those specific services but not about services the clients provide. The filter applied is IN. As a result, the following actions are applicable:
  - Allow every query IN.
  - Allow specific services in (such as printer services [IPP]).
  - Deny everything else.

In addition, to keep the service PTRs fresh in the cache an active query is configured. The active query queries for those services that we want to extend. Typically, this would match the services that have been configured as 'permitted' services in the IN filter. The value is set to 900 seconds. The duration is enough to refresh the PTRs as they typically have a TTL of 4500 seconds.

Example: Creating a Service Instance

```
Device> enable
Device# configure terminal
Device(config)# service-instance mdns-sd service printer-3 regtype _ipp._tcp.local domain
tcp4
Device(config-mdns-sd-si)# ipv4addr 209.165.200.230 255.255.255.0
Device(config-mdns-sd-si)# port 9100
Device(config-mdns-sd-si)# target-hostname fqdn-of-printer.example.com.
Device(config-mdns-sd-si)# txt _ipp._tcp.local=printer3
Device(config-mdns-sd-si)# priority 3
Device(config-mdns-sd-si)# weight 20
```

```
Device(config-mdns-sd-si)# exit
```

Note When you create a service-instance, a text record is created even if you do not configure service-instance parameters.

Additional References for Service Discovery Gateway

Related Documents

Related Topic	Document Title
Master Command List	Cisco IOS Master Command List
IP Addressing Services Command Reference	Cisco IOS IP Addressing Services Command Reference
Configuring DNS	IP Addressing: DNS Configuration Guide
DNS conceptual information	“Information About DNS” section in IP Addressing: DNS Configuration Guide

Standards and RFCs

Standard/RFC	Title
RFC 6762	Multicast DNS
RFC 6763	DNS-Based Service Discovery
Multicast DNS Internet-Draft	Multicast DNS Internet draft

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for Service Discovery Gateway

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 2: Feature Information for Service Discovery Gateway

Feature Name	Releases	Feature Information
Service Discovery Gateway	15.4(1)S	The Service Discovery Gateway feature enables multicast Domain Name System (mDNS) to operate across L3 boundaries (different subnets). The following commands were introduced or modified: **cache-memory-max, clear mdns cache, clear mdns statistics, debug mdns, match message-type, match service-instance, match service-type, redistribute mdns-sd, service-list mdns-sd, service-policy, service-policy-query, service-routing mdns-sd, show mdns cache, show mdns requests, show mdns statistics**
Service Discovery Gateway—Phase 2	15.5(2)S	The Service Discovery Gateway feature was enhanced with additional filter and configuration options. The following commands were introduced or modified: **clear mdns cache, clear mdns service-types, clear mdns statistics, designated-gateway, match location, rate-limit, service-instance mdns-sd, service-policy-proximity, service-routing mdns-sd, service-type-enumeration, show mdns cache, show mdns statistics, source-interface**

Feature Name	Releases	Feature Information
Service Discovery Gateway—Phase 3	15.5(2)S	The Service Discovery Gateway feature was enhanced with the following features: • **De-congestion of incoming mDNS traffic using the rate limiting mechanism—**The rate-limit value range was reset to1-100 p/s. • **Redistribution of service-withdrawal announcements across subnets when services are withdrawn, to improve mDNS cache efficiency and to avoid message loops—**The **withdraw-only** option was added to the **redistribute mdns-sd** command. • **A filter criterion for services available and learnt on a specific interface—**The **match learnt-interface** command was added to filter services. • **Enabling and disabling of periodic browsing of services on specific interfaces—**The **service-policy-query (interface)** command was added. For existing, globally configured active queries, the **disable** option was added to disable browsing of services on an interface, retaining the configurations on other interfaces. The following commands were introduced or modified: **match learnt-interface, rate-limit, redistribute mdns-sd, service-policy-query (interface)**