# IP Addressing: DNS Configuration Guide, Cisco IOS Release 15E

**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 **Service Discovery Gateway** 1

Information About Service Discovery Gateway 2

Filtering 2

Redistribution 3

How to Configure Service Discovery Gateway 3

Creating a Service-list, Applying a Filter for the Service-List and Configuring Parameters for the Service-List Name 3

Enabling mDNS Gateway for a Device 4

Applying a Service Policy 6

Verifying and troubleshooting Service Discovery Gateway 7

Configuration Examples for Service Discovery Gateway 8

Example: Creating a Service-List, Applying a Filter for the Service-List and Configuring Parameters for the Service-List Name 8

Example: Enabling mDNS Gateway for a Device 9

Example: Applying a Service Policy 9

Example: Creating Service Lists and Applying Service Policies 9

Additional References for Service Discovery Gateway 11

Feature Information for Service Discovery Gateway 12

CHAPTER 2 **VRF-Aware DNS** 15

Finding Feature Information 15

Information About VRF-Aware DNS 16

Domain Name System 16

VRF Mapping and VRF-Aware DNS 16

How to Configure VRF-Aware DNS 17

Defining a VRF Table and Assigning a Name Server to Enable VRF-Aware DNS 17

Mapping VRF-Specific Hostnames to IP Addresses 18

Configuring a Static Entry in a VRF-Specific Name Cache 19

Verifying the Name Cache Entries in the VRF Table 20
Configuration Examples for VRF-Aware DNS 21
Example: VRF-Specific Name Server Configuration 21
Example: VRF-Specific Domain Name List Configuration 21
Example: VRF-Specific Domain Name Configuration 22
Example: VRF-Specific IP Host Configuration 22
Additional References 22
Feature Information for VRF-Aware DNS 23

CHAPTER 1

# Service Discovery Gateway

The Service Discovery Gateway feature enables multicast Domain Name System (mDNS) to operate across Layer 3 (L3) boundaries (different subnets). An mDNS gateway will be able to provide transport for service discovery across Layer 3 boundaries by filtering, caching and extending services from one L3 domain (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.

### Service Announcement Redistribution

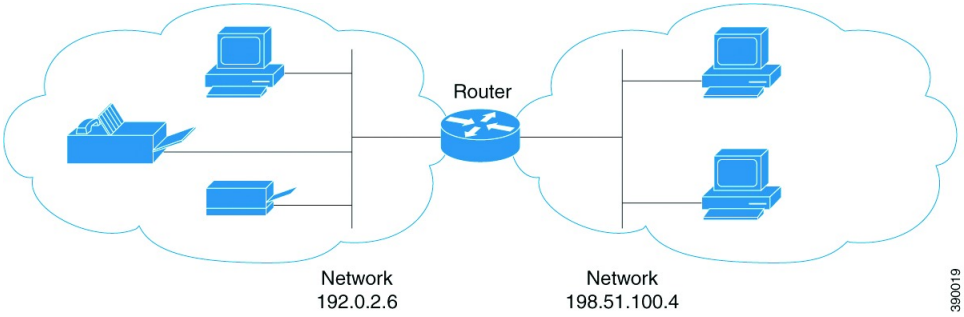
Service Extension usually works fine without actual replication of service announcements. The Service Discovery Gateway will cache announcements, queries and their responses in the cache. If another device queries for a service from a different subnet, the SDG will be able to provide an answer from its 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 or service announcement information cannot be done globally. You can enable redistribution of service information at the interface level only.


- Information About Service Discovery Gateway, page 2
- How to Configure Service Discovery Gateway, page 3
- Verifying and troubleshooting Service Discovery Gateway, page 7
- Configuration Examples for Service Discovery Gateway, page 8
- Additional References for Service Discovery Gateway , page 11
- Feature Information for Service Discovery Gateway , page 12

# Information About Service Discovery Gateway




You need to enable an mDNS gateway for service discovery to operate across subnets. You can enable mDNS gateway for a device or for an interface. You need to configure service routing globally before configuring at the interface level. After the device or interface is enabled, you can redistribute service discovery information across subnets. Also, you can create service policies and apply filters on either incoming service discovery information (called IN-bound filtering) or outgoing service discovery information (called OUT-bound filtering). Filters can be applied at the global level and at the interface level.

## Filtering

You can filter services that you want to extend selectively. While creating a service-list, the permit or deny option must be used. The permit option allows you to permit/transport specific service-list information. The deny option allows you to deny service-list information that is available to be transported to other subnets. You need to mention a sequence number when using the permit or deny option. The same service-list name can be associated with multiple sequence numbers and each sequence number will be mapped to a rule.

**Note** If no filters are configured, the default action is to deny service–list information to be transported through the device or interface.

Query is another option provided while creating service–lists. You can create queries using a service–list. If you want to browse for a service, then active queries can be used. This helps to keep the records refreshed in the cache.

Service-lists of type 'query' are used for active queries. Active queries will periodically send out requests for the given service names on all interfaces configured for service routing. As services have a specific TTL (Time to Live), this can help to keep services fresh in the cache.

**Note** Active queries can only be used globally and cannot be used at the interface level.

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 respond to queries.

After creating a service–list and using the permit or deny option, you can filter by using match statements (commands) based on service–instance, service–type, or message–type (announcement or query).

# Redistribution

**Note** Redistribution must be done selectively, and at the interface level only. Redistribution cannot be done globally.

Redistribution of Service Announcements is only required in specific scenarios. Generally, services like printers or Apple TV can be extended without any Service Announcement replication. 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.

# How to Configure Service Discovery Gateway

## Creating a Service-list, Applying a Filter for the Service-List and Configuring Parameters for the Service-List Name

### 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** } **OR match service-instance** {*instance-name* | **any** | **query**} **OR match service-type** *mDNS-service-type-string*
5. **exit**

### DETAILED STEPS

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

| | Command or Action | Purpose |
|---|---|---|
| **Step 3** | **service-list mdns-sd** *service-list-name* {**deny** *sequence-number* \| **permit** *sequence-number* \| **query**}<br><br>**Example:**<br>`Device(config)# service-list mdns-sd sl1 permit 3`<br>Or<br>`Device(config)# service-list mdns-sd sl4 query` | Enters mdns service discovery service-list mode.<br>• Creates a service-list and applies a filter on the service-list according to the **permit** or **deny** option applied to the sequence number.<br>Or<br>• Creates a service-list and associates a query for the service-list name if the **query** option is used. |
| **Step 4** | **match message-type** {**announcement** \| **any** \| **query** } **OR match service-instance** {*instance-name* \| **any** \| **query**} **OR match service-type** *mDNS-service-type-string*<br><br>**Example:**<br>Do one of the following:<br>`Device(config-mdns-sd-sl)# match message-type announcement`<br>OR<br>`Device(config-mdns-sd-sl)# match service-instance servInst 1`<br>OR<br>`Device(config-mdns-sd-sl)# match service-type _ipp._tcp` | Use one (or more) of the following commands.<br>Configures parameters for a service-list name that is created using step 3.<br>**Note** You cannot use the **match** command if you have used the **query** option in the previous step. The **match** command can be used only for the **permit** or **deny** option. |
| **Step 5** | **exit**<br><br>**Example:**<br>`Device(config-mdns-sd-sl)# exit` | Exits mdns service discovery service-list mode, and returns to global configuration mode. |

# Enabling mDNS Gateway for a Device

After enabling 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. You can set some part of the system memory for cache using the **cache-memory-max** command.

**Note** Steps 4 to 6 are optional and not meant to be used in any specific order.

## 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-query-name service-list-query-period*
7. **exit**

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br><br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br><br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **service-routing mdns-sd**<br><br>**Example:**<br>`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**}<br><br>**Example:**<br>`Device(config-mdns)# service-policy serv-pol1 IN` | For a service-list, applies a filter on incoming service discovery information (IN-bound filtering) or outgoing service discovery information (OUT-bound filtering).<br>**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*<br><br>**Example:**<br>`Device(config-mdns)# cache-memory-max 20` | Sets some part of the system memory (in percentage) for cache.<br>**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-query-name service-list-query-period*<br><br>**Example:**<br>`Device(config-mdns)# service-policy-query sl-query1 100` | Configures service-list-query period. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 7** | **exit**<br><br>**Example:**<br><br>`Device(config-mdns)# exit` | Exits multicast DNS configuration mode, and returns to global configuration mode. |

# Applying a Service Policy

## SUMMARY STEPS

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

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br><br>**Example:**<br><br>`Device> enable` | Enables privileged EXEC mode.<br><br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br><br>**Example:**<br><br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **service-routing mdns-sd**<br><br>**Example:**<br><br>`Device(config)# service-routing mdns-sd` | Enables mDNS gateway functionality for a device and enters multicast DNS configuration (config-mdns) mode. |
| **Step 4** | **interface** *type number*<br><br>**Example:**<br><br>`Device(config-mdns)# interface ethernet 0/1` | Enters Interface multicast DNS configuration mode, and enables interface configuration. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 5** | **service-policy** *service-policy-name* {**IN** \| **OUT**}<br><br>**Example:**<br><br>`Device(config-if-mdns)# service-policy serv-pol2 IN` | For a service-list, applies a filter on incoming service discovery information (IN-bound filtering) or outgoing service discovery information (OUT-bound filtering). |
| **Step 6** | **exit**<br><br>**Example:**<br><br>`Device(config-if-mdns)# exit` | Exits Interface multicast DNS configuration mode, and returns to multicast DNS 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** | *name record-name* | **type record-type** [*name record-name*]]
2. **show mdns cache** [**interface** *type number* | **name** *record-name*[**type** *record-type*] | **type** *recod-type*]
3. **show mdns statistics** {**all** | **service-list***list-name* | **service-policy** {**all** | **interface** *type number*}}
4. **debug mdns** {**all error event packet verbose**}

## DETAILED STEPS

**Step 1** **show mdns requests** [**detail** | *name record-name* | **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 mDNS requests, including record name and record type information.

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

**Example:**

```
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    smuchala-WS.local

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

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

This command displays mDNS cache information.

**Step 3** **show mdns statistics** {**all** | **service-list***list-name* | **service-policy** {**all** | **interface** *type number*}}

**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** **debug mdns** {**all error event packet verbose**}

**Example:**

```
Device# debug mdns
```

This command enables all mDNS debugging flows.

# Configuration Examples for Service Discovery Gateway

## Example: Creating a Service-List, Applying a Filter for the Service-List and Configuring Parameters for the Service-List Name

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)# exit
```

# Example: Enabling mDNS Gateway for a Device

The following example shows how to enable an mDNS gateway for a device. IN-bound filtering is applied on the service-list serv-pol1. 20% of system memory is made available for cache, and the service-list-query period is configured at 100 seconds.

```
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)# exit
```

# Example: Applying a Service Policy

```
Device> enable
Device# configure terminal
Device(config)# service-routing mdns-sd
Device(config-mdns)# interface ethernet 0/1
Device(config-if-mdns)# service-policy servpol2 IN
Device(config-if-mdns)# exit
```

# Example: Creating Service Lists and Applying Service Policies

The following example shows creation of service-lists mixed, permit-most, permit-all, and deny-all. Then, a service-policy is appropriately 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
!
service-list mdns-sd mixed permit 30
 match message-type announcement
 match service-type _ipp._tcp
!
service-list mdns-sd mixed permit 40
 match message-type announcement
 match service-type _airplay._tcp
!
service-list mdns-sd mixed deny 50
!
!
service-list mdns-sd permit-most deny 10
 match service-type _sleep-proxy._udp.
!
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
 service-type _ipp._tcp
 service-type _ipps._tcp
 service-type _raop._tcp
 service-type _airplay._tcp
!
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 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 AirPlay and 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.

# 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 |

### 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:<br>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.<br>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.<br>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 1: Feature Information for Service Discovery Gateway***

| Feature Name | Releases | Feature Information |
|---|---|---|
| Service Discovery Gateway | 15.2(1)E | The Service Discovery Gateway feature enables multicast Domain Name System (mDNS) to operate across L3 boundaries (different subnets).<br>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** |

CHAPTER 2

# VRF-Aware DNS

The VRF-Aware DNS feature enables the configuration of a Virtual Private Network (VPN) routing and forwarding instance (VRF) table so that the domain name system (DNS) can forward queries to name servers using the VRF table rather than the named DNS server in the global IP address space. This feature allows DNS requests to be resolved within the appropriate Multiprotocol Label Switching (MPLS) VPN.

**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, page 15
- Information About VRF-Aware DNS, page 16
- How to Configure VRF-Aware DNS, page 17
- Configuration Examples for VRF-Aware DNS, page 21
- Additional References, page 22
- Feature Information for VRF-Aware DNS, page 23


## 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.

# Information About VRF-Aware DNS

## Domain Name System

Domain Name System (DNS) is a standard that defines a domain naming procedure used in TCP/IP. A domain is a hierarchical separation of the network into groups and subgroups with domain names identifying the structure. The named groups consist of named objects, usually devices like IP hosts, and the subgroups are domains. DNS has three basic functions:

- Name space: This function is a hierarchical space organized from a single root into domains. Each domain can contain device names or more specific information. A special syntax defines valid names and identifies the domain names.
- Name registration: This function is used to enter names into the DNS database. Policies are outlined to resolve conflicts and other issues.
- Name resolution: This function is a distributed client and server name resolution standard. The name servers are software applications that run on a server and contain the resource records (RRs) that describe the names and addresses of those entities in the DNS name space. A name resolver is the interface between the client and the server. The name resolver requests information from the server about a name. A cache can be used by the name resolver to store learned names and addresses.

A DNS server can be a dedicated device or a software process running on a device. The server stores and manages data about domains and responds to requests for name conflict resolutions. In a large DNS implementation, there can be a distributed database over many devices. A server can be a dedicated cache.

## VRF Mapping and VRF-Aware DNS

To keep track of domain names, IP has defined the concept of a name server, whose job is to hold a cache (or database) of names appended to IP addresses. The cached information is important because the requesting DNS will not need to query for that information again, which is why DNS works well. If a server had to query each time for the same address because it had not saved any data, the queried servers would be flooded and would crash.

A gateway for multiple enterprise customers can be secured by mapping the remote users to a VRF domain. Mapping means obtaining the IP address of the VRF domain for the remote users. By using VRF domain mapping, a remote user can be authenticated by a VRF domain-specific AAA server so that the remote-access traffic can be forwarded within the VRF domain to the servers on the corporate network.

To support traffic for multiple VRF domains, the DNS and the servers used to resolve conflicts must be VRF aware. VRF aware means that a DNS subsystem will query the VRF name cache first, then the VRF domain, and store the returned RRs in a specific VRF name cache. Users are able to configure separate DNS name servers per VRF.

VRF-aware DNS forwards queries to name servers using the VRF table. Because the same IP address can be associated with different DNS servers in different VRF domains, a separate list of name caches for each VRF is maintained. The DNS looks up the specific VRF name cache first, if a table has been specified, before sending a query to the VRF name server. All IP addresses obtained from a VRF-specific name cache are routed using the VRF table.

# How to Configure VRF-Aware DNS

## Defining a VRF Table and Assigning a Name Server to Enable VRF-Aware DNS

Perform this task to define a VRF table and assign a name server.

A VRF-specific name cache is dynamically created if one does not exist whenever a VRF-specific name server is configured by using the **ip name-server vrf**command option or a permanent name entry is configured by using the **ip host vrf**command option. The VRF name cache is removed whenever all name server and permanent entries in the VRF are disabled.

It is possible that multiple name servers are configured with the same VRF name. The system will send queries to those servers in turn until any of them responds, starting with the server that sent a response the last time.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip vrf** *vrf-name*
4. **rd** *route-distinguisher*
5. **exit**
6. **ip name-server** [**vrf** *vrf-name*] *server-address1* [*server-address2...server-address6*]
7. **ip domain lookup** [**source-interface** *interface-type interface-number*]

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br><br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br><br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip vrf** *vrf-name*<br><br>**Example:**<br>`Device(config)# ip vrf vpn1` | Defines a VRF table and enters VRF configuration mode.<br>• The *vrf-name* argument can be up to 32 characters. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 4** | **rd** *route-distinguisher*<br><br>**Example:**<br><br>`Device(config)# rd 100:21` | Creates routing and forwarding tables for a VRF. |
| **Step 5** | **exit**<br><br>**Example:**<br><br>`Device(config-vrf)# exit` | Exits VRF configuration mode. |
| **Step 6** | **ip name-server** [**vrf** *vrf-name*] *server-address1* [*server-address2...server-address6*]<br><br>**Example:**<br><br>`Device(config)# ip name-server vrf vpn1 172.16.1.111 2001:DB8:1::1` | Assigns the address of one or more name servers to a VRF table to use for name and address resolution.<br><br>• The name server IP address can be an IPv4 or IPv6 address.<br><br>• The **vrf** keyword is optional but must be specified if the name server is used with VRF. The *vrf-name* argument assigns a name to the VRF. |
| **Step 7** | **ip domain lookup** [**source-interface** *interface-type interface-number*]<br><br>**Example:**<br><br>`Device(config)# ip domain lookup` | (Optional) Enables DNS-based address translation.<br><br>• DNS is enabled by default. You only need to use this command if DNS has been disabled. |

# Mapping VRF-Specific Hostnames to IP Addresses

Perform this task to map VRF-specific hostnames to IP addresses.

## SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. Do one of the following:
   - **ip domain name** [**vrf** *vrf-name*] *name*
   - 
   - **ip domain list** [**vrf** *vrf-name*] *name*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br><br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br><br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | Do one of the following:<br>• **ip domain name** [**vrf** *vrf-name*] *name*<br>•<br>• **ip domain list** [**vrf** *vrf-name*] *name*<br><br>**Example:**<br>`Device(config)# ip domain name vrf vpn1 cisco.com`<br><br>**Example:**<br>`Device(config)# ip domain list vrf vpn1 cisco.com` | Defines a default domain name that the software will use to complete unqualified hostnames.<br>or<br>Defines a list of default domain names to complete unqualified hostnames.<br>• You can specify a default domain name that the 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.<br>• The **vrf** keyword and *vrf-name* argument specify a default VRF domain name.<br>• The **ip domain list** command can be entered multiple times to specify more than one domain name to append when doing a DNS query. The system will append each in turn until it finds a match. |

# Configuring a Static Entry in a VRF-Specific Name Cache

Perform this task to configure a static entry in a VRF-specific name cache.

A VRF-specific name cache is dynamically created if one does not exist whenever a name server is configured for the VRF by using the **ip name-server vrf**command option or a permanent name entry is configured by using the **ip host vrf** command option. The VRF name cache is removed whenever all name server and permanent entries in the VRF are disabled.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip host** [**vrf** *vrf-name*] *name* [*tcp-port*] *address1* [*address2...address8*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br><br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br><br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip host** [**vrf** *vrf-name*] *name* [*tcp-port*] *address1* [*address2...address8*<br><br>**Example:**<br><br>**Example:**<br>`Device(config)# ip host vrf vpn3 company1.com 172.16.2.1` | Defines a static hostname-to-address mapping in the host cache.<br>• The IP address of the host can be an IPv4 or IPv6 address, and the IP address can be associated with a Virtual Private Network (VPN) routing and forwarding (VRF) instance.<br>• If the **vrf** keyword and *vrf-name* arguments are specified, then a permanent entry is created only in the VRF-specific name cache. |

# Verifying the Name Cache Entries in the VRF Table

Perform this task to verify the name cache entries in the VRF table.

### SUMMARY STEPS

1. **enable**
2. **show hosts** [**vrf** *vrf-name*] {**all**| *hostname*} [**summary**]
3. **clear host** [**vrf** *vrf-name*] {**all**| *hostname*}

### DETAILED STEPS

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

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | **show hosts** [**vrf** *vrf-name*] {**all**\| *hostname*} [**summary**]<br><br>**Example:**<br><br>`Device# show hosts vrf vpn2` | • Displays the default domain name, the style of name lookup service, a list of name server hosts, the cached list of hostnames and addresses, and the cached list of hostnames and addresses specific to a particular Virtual Private Network (VPN).<br>• The **vrf** keyword and *vrf-name* argument only display the entries if a VRF name has been configured.<br>• If you enter the **show hosts** command without specifying any VRF, only the entries in the global name cache will display. |
| **Step 3** | **clear host** [**vrf** *vrf-name*] {**all**\| *hostname*}<br><br>**Example:**<br><br>`Device# clear host vrf vpn2` | (Optional) Deletes entries from the hostname-to-address global address cache or VRF name cache. |

# Configuration Examples for VRF-Aware DNS

## Example: VRF-Specific Name Server Configuration

The following example shows how to specify a VPN named vpn1 with the IP addresses of 172.16.1.111 and 172.16.1.2 as the name servers:

```
ip name-server vrf vpn1 172.16.1.111 172.16.1.2
```

## Example: VRF-Specific Domain Name List Configuration

The following example shows how to add several domain names to a list in vpn1 and vpn2. The domain name is only used for name queries in the specified VRF.

```
ip domain list vrf vpn1 company.com
ip domain list vrf vpn2 school.edu
```

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 a match is found.

## Example: VRF-Specific Domain Name Configuration

The following example shows how to define cisco.com as the default domain name for a VPN named vpn1. The domain name is only used for name queries in the specified VRF.

```
ip domain name vrf vpn1 cisco.com
```

Any IP hostname that does not contain a domain name (that is, any name without a dot) will have the dot and cisco.com appended to it before being looked up.

## Example: VRF-Specific IP Host Configuration

The following example shows how to define two static hostname-to-address mappings in the host cache for vpn2 and vpn3:

```
ip host vrf vpn2 host2 10.168.7.18
ip host vrf vpn3 host3 10.12.0.2
```

# Additional References

### Related Documents

| Related Topic | Document Title |
|---|---|
| DNS configuration tasks | "Configuring DNS" module |
| IP addressing services 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 feature, and support for existing standards has not been modified by this feature. | -- |

### 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:<br>http://www.cisco.com/go/mibs |

### RFCs

| RFCs | Title |
|---|---|
| No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature. | -- |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for VRF-Aware 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 2: Feature Information for VRF-Aware DNS*

| Feature Name | Releases | Feature Information |
|---|---|---|
| VRF-Aware DNS | 12.4(4)T<br>15.2(1)E<br>15.4(1)T | The VRF-Aware DNS feature enables the configuration of a Virtual Private Network (VPN) routing and forwarding instance (VRF) table so that the domain name system (DNS) can forward queries to name servers using the VRF table rather than the named DNS server in the global IP address space. This feature allows DNS requests to be resolved within the appropriate Multiprotocol Label Switching (MPLS) VPN. |