co – Filtering Routing Updates on Distance Vector IP Routing P

Table of Contents

Filtering Routing Updates on Distance Vector IP Routing Protocols

Contents

Introduction

This document explains various methods of filtering routes and the effects of applying the filters. The filters covered in this document are ones that prevent updates through router interfaces, ones that control the advertising of routes in routing updates, and ones that control the processing of routing updates.

Because route filtering works by regulating the routes that are entered into or advertised out of the route table, they have different effects on link state routing protocols than they do on distance vector protocols. A router running a distance vector protocol advertises routes based on what is in its route table. As a result, a route filter influences which routes the router advertises to its neighbors.

On the other hand, routers running link state protocols determine their routes based on information in their link state database, rather than on the advertised route entries of its neighbors. Route filters have no effect on links state advertisements or on the link state database. For this reason, the information in this document only applies to distance vector IP Routing Protocols such as Routing Information Protocol (RIP), RIP version 2, Interior Gateway Routing Protocol (IGRP), and Enhanced IGRP (EIGRP).

Preventing Routing Updates Through an Interface

Using the **passive interface** command can prevent routers from sending routing updates through a router interface. Keeping routing update messages from being sent through a router interface prevents other systems on that network from learning about routes dynamically. For examples using the **passive interface** command, see the "Passive Interface Examples" section in Configuring IP Routing Protocol−Independent Features.

For RIP and IGRP, the **passive interface** command stops the router from sending updates to a particular neighbor, but the router continues to listen and use routing updates from that neighbor; however, on EIGRP, the **passive interface** command affects the protocol differently, as explained in How Does the Passive Interface Feature Work in EIGRP?

Controlling the Processing and Advertising of Routes in Routing Updates

To control the advertising and processing of routes in routing updates, use the **distribute−list** command. There are two **distribute−list** commands: **distribute−list in** and **distribute−list out**. They are similar in syntax, but the options available to each and their behavior are very different.

The **distribute−list in** command is used to control which routes are processed in incoming routing updates. See below for an example of this command.

The **distribute−list out** command is used to control which routes are included in outgoing routing updates. See the Using `distribute−list out` section for an example.

Using `distribute−list in`

The syntax for the **distribute−list in** command is:

distribute−list *access−list−number* **in** [*interface−name*]

where *access−list−number* is the standard IP access−list against which the contents of the incoming routing update are matched. The [*interface−name*] argument is optional and specifies the interface on which the update is expected. It is important to note that the access−list referred to in *access−list−number* is applied to the contents of the update, not to the source or destination of the routing update packets. The router decides whether or not to include the contents in its routing table based on the access−lists. For example:

```
access−list 1 permit 1.0.0.0 0.255.255.255
router rip
distribute−list 1 in
```

Any inbound RIP update is checked against **access−list 1** and only routes that match a **1.xxx.xxx.xxx** format are put in the routing table.

For a given routing process, it is possible to define one inbound interface−specific distribute−list per interface, and one globally−defined distribute−list. For example, the following combination is possible:

```
access−list 1 permit 1.0.0.0 0.255.255.255
access−list 2 permit 1.2.3.0 0.0.0.255
router rip
distribute−list 2 in ethernet 0
distribute−list 1 in
```

In this scenario, the router checks the interface on which the update comes in. If it is Ethernet 0, **access−list 2** is applied before putting it in the routing table. If, based on this check, the network is denied, no further checking is done. However, if **distribute−list 2** allows the network, then distribute−list 1 is also checked. If both distribute−lists allow the network, it is put in the table. The following algorithm is followed when multiple distribute−lists are used.

1. Extract the next network from the inbound update.
2. Check the interface that it came into.
3. Is there a distribute list applied to that interface?
    - Yes: Is the network denied by that list?

◊ Yes: the network does not make it; return to step 1

◊ No: the network is allowed; continue to step 4.

♦ No: Go to step 4.

4. Is there a global distribute list?

♦ Yes: Is the network denied by that list?

◊ Yes: the network does not make it; return to step 1.

◊ No: the network makes it; return to step 1.

♦ No: The network makes it; return to step 1.

Using `distribute−list out`

The syntax for the **distribute−list out** command is:

distribute−list *access−list−number* **out** [*interface−name*|*routing process*|*autonomous−system−number*]

where *access−list−number* is the standard IP access−list against which the contents of the outgoing routing updates are matched. The [interface−name] argument is optional, and specifies on which interface the update is going out. The [*routing process*|*autonomous−system−number*] arguments are used when redistribution from another routing process or autonomous system number has been specified. The list is applied to any routes imported from the specified process into the current one.

For example:

```
access−list 1 permit 1.0.0.0 0.255.255.255
router rip
default−metric 1
redistribute igrp 20
distribute−list 1 out igrp 20
```

Here, routes from **igrp 20** are being redistributed into RIP. Any outbound routing update that was originally sourced from **igrp 20** is checked against **access−list 1**. Only routes that match a **1.xxx.xxx.xxx** format are sent.

Note that it is possible to specify multiple distribute−lists for a given routing process if they are applied to different interfaces, or globally. For any given routing protocol, it is possible to define one interface−specific distribute−list per interface and one protocol−specific distribute−list for each process/autonomous−system pair.

```
access−list 1 permit 1.0.0.0 0.255.255.255
access−list 2 permit 1.2.3.0 0.0.0.255
router rip
distribute−list 2 out ethernet 0
distribute−list 1 out
```

In this scenario, the router only sends routes pertaining to the 1.2.3.0 subnet out of Ethernet 0, and any updates about networks in the 1.0.0.0 are flooded out to the remaining interfaces, including the 1.2.3.0 subnet. The following algorithm is used when multiple distribute−lists are used.

1. Select the next network to receive an outbound update.
2. Check which interface it is being sent out on.
3. Is there a distribute list applied to that interface?
    - ♦ Yes: Is the network denied by that list?
        - ◊ Yes: the network does not go out; return to step 1.
        - ◊ No: the network goes out; continue to step 4.
    - ♦ No: Go to step 4.
4. Check the routing process or AS from which we derive the route.
5. Is there a distribute list applied to that process or AS?
    - ♦ Yes: Is the network denied by that list?
        - ◊ Yes: the network does not go out; return to step 1.
        - ◊ No: the network goes out; continue to step 6.
    - ♦ No: Go to step 6.
6. Is there a global distribute list?
    - ♦ Yes: Is the network denied by that list?
        - ◊ Yes: the network does not go out; return to step 1.
        - ◊ No: the network goes out; return to step 1.
    - ♦ No: The network makes it; go to step 1.

Note that checking the distribute list is only one of the many checks that are done against a distance vector route before a router includes it in the routing table or in an update. Checks are also made for desirability, policies, split horizon, and other factors.

Related Information

- **Routing Protocol Technical Tips**
- **IP Routing Protocols Top Issues**
- **IP Routing Protocols Support Pages**

All contents are Copyright © 1992−−2001 Cisco Systems Inc. All rights reserved. Important Notices and Privacy Statement.