DMVPN NHRP Event Publisher

The DMVPN: NHRP Event Publisher feature allows you to publish Next Hop Resolution Protocol (NHRP) specific events to the Event Detector (ED). NHRP publishes NHRP events with data to the NHRP-ED handler. The DMVPN: NHRP Event Publisher feature enhances Dynamic Multipoint VPN (DMVPN) with the capability to control the building of dynamic spoke-to-spoke tunnels. This feature also optimizes the conditions under which spokes build dynamic tunnels with each other. It also integrates Embedded Event Manager (EEM) with NHRP and leverages EEM scripts to influence the behavior of NHRP. In this feature, the only event that is supported is the capability to build dynamic spoke-to-spoke tunnels.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see 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 document.

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 DMVPN NHRP Event Publisher

You need to use the **nhrpevent-publishermax-event-timeout** command to turn on the DMVPN: NHRP Event Publisher feature. For information on DMVPN configuration, see Configuring Dynamic Multipoint VPN . For information on NHRP configuration, see Configuring NHRP .

Restrictions for DMVPN NHRP Event Publisher

You cannot manually configure spoke-to-spoke tunneling with this feature. You can only build dynamic spoke-to-spoke tunnels.

Information About DMVPN NHRP Event Publisher

Dynamic Spoke-to-Spoke Tunnels

Spoke-to-spoke tunnels are designed to be dynamic, in that they are created only when there is data traffic that uses the tunnel; and they are removed when there is no data traffic using the tunnel.

In addition to NHRP registration of next hop clients (NHCs) with next hop servers (NHSs), NHRP provides the capability for NHCs (spokes) to find a shortcut path over the infrastructure of the network (IP network, Switched Multimegabit Data Service [SMDS]) or to build a shortcut switched virtual circuit (SVC) over a switched infrastructure network (Frame Relay and ATM) directly to another NHC (spoke), bypassing hops through the NHSs (hubs). This capability allows the building of very large NHRP-NBMA networks. In this way, the bandwidth and CPU limitations of the hub do not limit the overall bandwidth of the NHRP-NBMA network. This capability effectively creates a full-mesh-capable network without having to discover all possible connections beforehand. This type of network is called a dynamic-mesh network, where there is a base hub-and-spoke network of NHCs and NHSs. The network of NHCs and NHSs is used for transporting NHRP, dynamic routing protocol information, data traffic, and dynamic direct spoke-to-spoke links. The spoke-to-spoke links are built when there is data traffic to use the link, and the spoke-to-spoke links are torn down when the data traffic stops.

The dynamic-mesh network allows individual spoke routers to directly connect to anywhere in the NBMA network, even though they are capable of connecting only to a limited number at the same time. This functionality allows each spoke in the network to participate in the whole network up to its capabilities without limiting another spoke from participating up to its capability. If a full-mesh network were to be built, all spokes would have to be sized to handle all possible tunnels at the same time.

For example, in a network of 1000 nodes, a full-mesh spoke would need to be large and powerful because it must always support 999 tunnels (one to every other node). In a dynamic-mesh network, a spoke needs to support only a limited number of tunnels to its NHSs (hubs) plus any currently active tunnels to other spokes. Also, if a spoke cannot build more spoke-to-spoke tunnels, it will send its data traffic by way of the spoke-hub-spoke path. This design ensures that connectivity is always preserved, even when the preferred single hop path is not available.

DMVPN NHRP Event Publisher

Currently DMVPN establishes a direct spoke-to-spoke tunnel with shortcut switching enabled on the spoke and NHRP redirect on the hub, without performing any additional checks before establishing traffic on the tunnel. This direct spoke-to-spoke tunnel may not be the best path as there could be other alternative best paths available for this traffic.

The DMVPN: NHRP Event Publisher feature performs additional checks before establishing the spoke-to-spoke tunnel and sending traffic on the tunnel. This feature helps the administrator to decide about the local policies and attributes while building the tunnel. This prevents known bad network connections based on local history

or centralized information. It also reduces the administrative overhead by monitoring available resources and selecting the best options.

Embedded Event Manager

Embedded Event Manager (EEM) is a powerful and flexible subsystem in Cisco IOS software that provides real-time network event detection and onboard automation. Using EEM, you can adapt the behavior of your network devices to align with your business needs. EEM is available on a wide range of Cisco platforms, and customers can benefit from the capabilities of EEM without upgrading to a new version of IOS.

EEM supports over 20 event detectors that are integrated with different Cisco IOS components to trigger actions in response to network events. Business logic can be injected into various networking operations using EEM policies. These policies are programmed using either a simple CLI-based interface or a scripting language called Tool Command Language (TCL). EEM harnesses the significant intelligence within Cisco devices to enable creative solutions including automated troubleshooting, automatic fault detection and troubleshooting, and device configuration automation.

EEM is implemented through the creation of policies. An EEM policy is an entity that defines an event and the actions to be taken when that event occurs. There are two types of EEM policies: an applet and a script. An applet is a simple form of policy that is defined within the CLI configuration. A script is a form of policy that is written in TCL. When an EEM policy is registered with the EEM, the software examines the policy and registers it to be run when the specified event occurs. Policies can be unregistered or suspended.

The following tasks are required to create an EEM policy:

- Selecting the event for which the policy is run.
- Defining the Event Detector (ED) options associated with logging and responding to the event.
- Defining the environment variables, if required.
- Choosing the actions to be performed when the event occurs.

NHRP Event Publishing Flow

When a local spoke sends a resolution request to a remote spoke, the remote spoke triggers the EEM. The EEM decides whether to connect to or reject the request. If the EEM agrees to connect, the remote spoke builds the tunnel and sends the resolution reply through the tunnel.

Making NHRP be the ED helps define your own events, and the application can create and publish these events. On the remote spoke, the TCL scripts can subscribe to these events. The published events are sent to the subscribed TCL scripts. NHRP events are published to the NHRP-ED handler. The event information is copied to the XML buffer, and the NHRP-ED publishes this buffer to the EEM server. The event subscriber (TCL scripts from the remote spoke) receives and registers the event request so that the remote spoke is notified when the event is published. The TCL script replies to NHRP with the **ipnhrpconnectreqid**or **ipnhrprejectreqid**command. The **ipnhrpconnectreqid** command enables the spoke to initiate a resolution reply for the received request to build a shortcut tunnel. The **ipnhrprejectreqid** command prevents the spoke from initiating the resolution reply for the received request.

The **ipnhrpconnectreqid** command invokes connect registry callback as an action to trigger the resolution reply. The remote spoke either builds the spoke-to-spoke tunnel and sends the resolution reply within the tunnel or sends the resolution reply with the policy attributes through the hub. If the resolution reply is sent through the hub, the spoke receiving the resolution reply builds the spoke-to-spoke tunnel.

When the TCL script responds with the **ipnhrprejectreqid** command, the remote spoke does not build the spoke-to-spoke tunnel. It sends the NHRP resolution NAK message with a reject time value and subnet mask to the local spoke through the hub.

The following sequence lists the NHRP event flow:

1. An NHRP event registers with the NHRP-ED.
2. The application creates an event definition.
3. A TCL script subscribes for NHRP event receipt asking that the script's callback routine be invoked when the event is published.
4. The NHRP ED detects an event and contacts the EEM at the remote spoke.
5. The EEM schedules the event processing calling the application's callback handler routine.
6. The TCL script returns the callback routine.

How to Configure DMVPN NHRP Event Publisher

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **tunnel mode gre multipoint**
5. **tunnel key** *key-number*
6. **ip nhrp network-id** *number*
7. **ip nhrp attribute set isp-name** *value*
8. **nhrp event timer**
9. **end**
10. **show ipv6 nhrp attribute**
11. **show ip nhrp attribute**
12. **show dmvpn detail**
13. **debug nhrp attribute**
14. **exit**

DETAILED STEPS

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

	Command or Action	Purpose
	`Router# configure terminal`	
Step 3	**interface** *type number* **Example:** `Router(config)# interface tunnel 100`	Configures an interface and enters interface configuration mode.
Step 4	**tunnel mode gre multipoint** **Example:** `Router(config-if)# tunnel mode gre multipoint`	Enables a GRE tunnel to be used in multipoint NBMA mode.
Step 5	**tunnel key** *key-number* **Example:** `Router(config-if)# tunnel key 3`	(Optional) Sets the tunnel ID key.
Step 6	**ip nhrp network-id** *number* **Example:** `Router(config-if)# ip nhrp network-id 1`	Enables NHRP on the interface.
Step 7	**ip nhrp attribute set isp-name** *value* **Example:** `Router(config-if)# ip nhrp attribute set isp-name 200`	Sets the local policy attributes that are carried in NHRP resolution requests.
Step 8	**nhrp event timer** **Example:** `Router(config-if)# nhrp event timer`	Publishes an NHRP event with the attributes to EEM.
Step 9	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 10	**show ipv6 nhrp attribute** **Example:** `Router# show ipv6 nhrp attribute`	Displays the IPv6 NHRP attributes configured on the spoke.
Step 11	**show ip nhrp attribute** **Example:** `Router# show ip nhrp attribute`	Displays the IP NHRP attributes configured on the spoke.

	Command or Action	Purpose
Step 12	**show dmvpn detail** **Example:** `Router# show dmvpn detail`	Displays DMVPN-specific session information.
Step 13	**debug nhrp attribute** **Example:** `Router# debug nhrp attribute`	Enables NHRP debugging.
Step 14	**exit** **Example:** `Router# exit`	Exits privileged EXEC mode.

What to do next

Configuration Examples for DMVPN NHRP Event Publisher

Example Configuring DMVPN NHRP Event Publisher

The following is a sample configuration of the DMVPN: NHRP Event Publisher feature:

```
interface tunnel 100
 tunnel mode gre multipoint
 tunnel key 3
 ip nhrp network-id 1
 ip nhrp attribute set isp-name 200
 nhrp event timer
 end
  show ip nhrp attribute
  show dmvpn detail
  debug nhrp attribute
```

Additional References

Related Documents

Related Topic	Document Title
Configuring Dynamic Multipoint VPN	Configuring Dynamic Multipoint VPN
Configuring NHRP	Configuring NHRP
NHRP commands	Cisco IOS IP Addressing Services Command Reference

RFCs

RFC	Title
RFC 2332	*NBMA Next Hop Resolution Protocol (NHRP)*

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 DMVPN NHRP Event Publisher

The following table lists the release history for this feature.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note The following table lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1: Feature Information for DMVPN: NHRP Event Publisher

Feature Name	Releases	Feature Information
DMVPN: NHRP Event Publisher	15.2(2)T	The DMVPN: NHRP Event Publisher feature allows you to publish NHRP specific events to the ED. This feature enhances DMVPN with the capability to control the ability to build dynamic spoke-to-spoke tunnels. This feature also optimizes the conditions under which spokes build dynamic tunnels with each other. It also integrates EEM with NHRP. The following commands were introduced or modified: **ipnhrpconnect**, **ipnhrpreject**, **showipnrhpattribute**, **showipv6nhrpattribute**.