Configuring SNMP Support

Simple Network Management Protocol (SNMP) is an application-layer protocol that provides a message format for communication between SNMP managers and agents. SNMP provides a standardized framework and a common language that is used for monitoring and managing devices in a network.

This document discusses how to enable an SNMP agent on a Cisco device and how to control the sending of SNMP notifications from the agent. For information about using SNMP management systems, see the appropriate documentation for your network management system (NMS) application.

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.

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 Configuring SNMP Support

Components of SNMP

The Simple Network Management Protocol (SNMP) is an application-layer protocol that provides a message format for communication between SNMP managers and agents. SNMP provides a standardized framework and a common language used for monitoring and managing devices in a network.

The SNMP framework has the following components, which are described in the following sections:

SNMP Manager

The Simple Network Management Protocol (SNMP) manager is a system that controls and monitors the activities of network hosts using SNMP. The most common managing system is a network management system (NMS). The term NMS can be applied either to a dedicated device used for network management or to the applications used on such a device. Several network management applications are available for use with SNMP and range from simple command line interface applications to applications such as the CiscoWorks2000 products that use GUIs.

SNMP Agent

The Simple Network Management Protocol (SNMP) agent is the software component within a managed device that maintains the data for the device and reports this data, as needed, to managing systems. The agent resides on the routing device (router, access server, or switch). To enable an SNMP agent on a Cisco routing device, you must define the relationship between the manager and the agent.

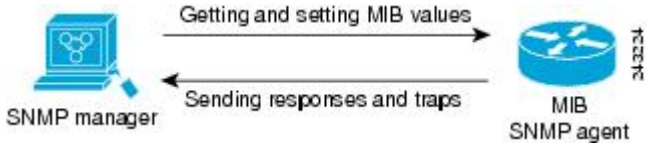
Note Although many Cisco devices can be configured to be an SNMP agent, this practice is not recommended. Commands that an agent needs to control the SNMP process are available through the Cisco command line interface without additional configuration.

SNMP MIB

An SNMP agent contains MIB variables, whose values the SNMP manager can request or change through Get or Set operations. A manager can get a value from an agent or store a value in that agent. The agent gathers data from the SNMP MIB, the repository for information about device parameters and network data. The agent can also respond to manager requests to get or set data.

The figure below illustrates the communications between the SNMP manager and agent. A manager sends an agent requests to get and set the SNMP MIB values. The agent responds to these requests. Independent of this interaction, the agent can send the manager unsolicited notifications (traps or informs) to notify the manager about network conditions.

Figure 1: Communication Between an SNMP Agent and Manager



SNMP Operations

The Simple Network Management Protocol (SNMP) applications perform the following operations to retrieve data, modify SNMP object variables, and send notifications:

SNMP Get

The Simple Network Management Protocol (SNMP) GET operation is performed by an Network Management Server (NMS) to retrieve SNMP object variables. There are three types of GET operations:

- GET—Retrieves the exact object instance from the SNMP agent.

- GETNEXT—Retrieves the next object variable, which is a lexicographical successor to the specified variable.
- GETBULK—Retrieves a large amount of object variable data, without the need for repeated GETNEXT operations.

SNMP SET

The Simple Network Management Protocol (SNMP) SET operation is performed by a Network Management Server (NMS) to modify the value of an object variable.

SNMP Notifications

A key feature of Simple Network Management Protocol (SNMP) is its capability to generate unsolicited notifications from an SNMP agent.

Traps and Informs

Unsolicited (asynchronous) notifications can be generated as traps or inform requests (informs). Traps are messages alerting the Simple Network Management Protocol (SNMP) manager to a condition on the network. Informs are traps that include a request for confirmation of receipt from the SNMP manager. Notifications can indicate improper user authentication, restarts, the closing of a connection, loss of connection to a neighbor device, or other significant events.

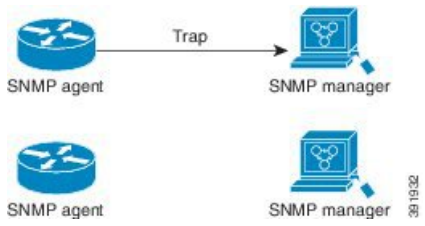
Traps are less reliable than informs because the receiver does not send an acknowledgment when it receives a trap. The sender does not know if the trap was received. An SNMP manager that receives an inform acknowledges the message with an SNMP response protocol data unit (PDU). If the sender never receives a response, the inform can be sent again. Thus, informs are more likely to reach their intended destination.

Traps are often preferred even though they are less reliable because informs consume more resources in the device and the network. Unlike a trap, which is discarded as soon as it is sent, an inform must be held in memory until a response is received or the request times out. Also, traps are sent only once, whereas an inform may be resent several times. The retries increase traffic and contribute to higher overhead on the network. Use of traps and informs requires a trade-off between reliability and resources. If it is important that the SNMP manager receives every notification, use informs. However, if traffic volume or memory usage are concerns and receipt of every notification is not required, use traps.

The figures below illustrate the differences between traps and informs.

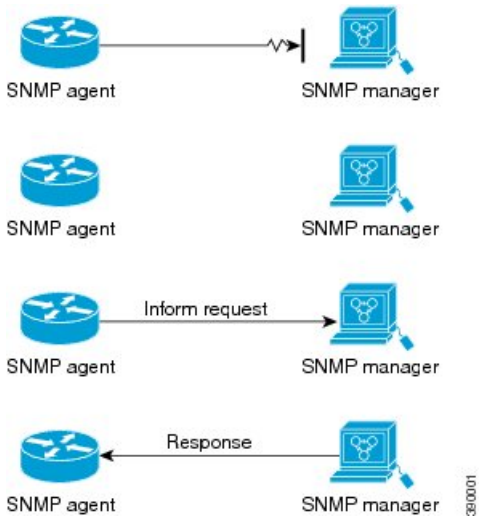
The figure below shows that an agent successfully sends a trap to an SNMP manager. Although the manager receives the trap, it does not send an acknowledgment. The agent has no way of knowing that the trap reached its destination.

Figure 2: Trap Successfully Sent to SNMP Manager



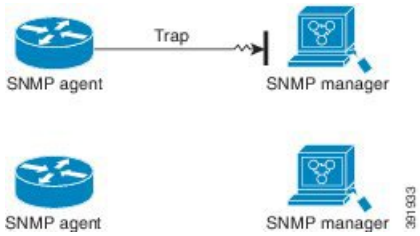
In the figure below, the agent successfully sends an inform to the manager. When the manager receives the inform, a response is sent to the agent, and the agent knows that the inform reached its destination. Note that in this example, the traffic generated is twice as much as in the interaction shown in the figure above.

Figure 3: Inform Request Successfully Sent to SNMP Manager



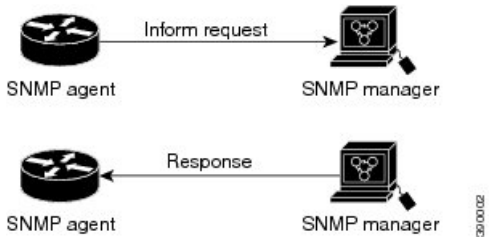
The figure below shows an agent sending a trap to a manager that the manager does not receive. The agent has no way of knowing that the trap did not reach its destination. The manager never receives the trap because traps are not resent.

Figure 4: Trap Unsuccessfully Sent to SNMP Manager



The figure below shows an agent sending an inform to a manager that does not reach the manager. Because the manager did not receive the inform, it does not send a response. After a period of time, the agent resends the inform. The manager receives the inform from the second transmission and replies. In this example, more traffic is generated than in the scenario shown in the figure above, but the notification reaches the SNMP manager.

Figure 5: Inform Unsuccessfully Sent to SNMP Manager



Note Whenever an SNMP process comes up, the reserved ports 161 and 162 are used. In addition to these two reserved ports, a dynamic port is also opened to run the SNMP proxy forwarder application.

MIBs and RFCs

MIB modules typically are defined in RFC documents submitted to the IETF, an international standards body. RFCs are written by individuals or groups for consideration by the Internet Society and the Internet community as a whole, usually with the intention of establishing a recommended Internet standard. Before being given RFC status, recommendations are published as Internet Draft (I-D) documents. RFCs that have become recommended standards are also labeled as standards documents (STDs). You can learn about the standards process and the activities of the IETF at the Internet Society website at http://www.isoc.org. You can read the full text of all RFCs, I-Ds, and STDs referenced in Cisco documentation at the IETF website at http://www.ietf.org.

The Cisco implementation of SNMP uses the definitions of MIB II variables described in RFC 1213 and definitions of Simple Network Management Protocol (SNMP) traps described in RFC 1215.

Cisco provides its own private MIB extensions with every system. Cisco enterprise MIBs comply with the guidelines described in the relevant RFCs unless otherwise noted in the documentation. You can find the MIB module definition files and the list of MIBs supported on each Cisco platform on the Cisco MIB website on Cisco.com.

Versions of SNMP

The Cisco IOS software supports the following versions of SNMP:

- SNMPv1—Simple Network Management Protocol: a full Internet standard, defined in RFC 1157. (RFC 1157 replaces the earlier versions that were published as RFC 1067 and RFC 1098.) Security is based on community strings.
- SNMPv2c—The community string-based Administrative Framework for SNMPv2. SNMPv2c (the “c” is for “community”) is an experimental Internet protocol defined in RFC 1901, RFC 1905, and RFC 1906. SNMPv2c is an update of the protocol operations and data types of SNMPv2p (SNMPv2 Classic) and uses the community-based security model of SNMPv1.
- SNMPv3—Version 3 of SNMP. SNMPv3 is an interoperable standards-based protocol defined in RFCs 3413 to 3415. SNMPv3 provides secure access to devices by authenticating and encrypting packets over the network.

The security features provided in SNMPv3 are as follows:

- Message integrity—Ensuring that a packet has not been tampered with in transit.
- Authentication—Determining that the message is from a valid source.
- Encryption—Scrambling the contents of a packet to prevent it from being learned by an unauthorized source.

Both SNMPv1 and SNMPv2c use a community-based form of security. The community of SNMP managers able to access the agent MIB is defined by a community string.

SNMPv2c support includes a bulk retrieval mechanism and detailed error message reporting to management stations. The bulk retrieval mechanism supports the retrieval of tables and large quantities of information, minimizing the number of round trips required. The SNMPv2c improved error handling support includes expanded error codes that distinguish different types of errors; these conditions are reported through a single error code in SNMPv1. The following three types of exceptions are also reported: no such object, no such instance, and end of MIB view.

SNMPv3 is a security model in which an authentication strategy is set up for a user and the group in which the user resides. A security level is the permitted level of security within a security model. A combination of a security model and a security level determines which security mechanism is employed when handling an SNMP packet.

Three security models are available: SNMPv1, SNMPv2c, and SNMPv3. The table below lists the combinations of security models and levels and their meanings.

Table 1: SNMP Security Models and Levels

Model	Level	Authentication	Encryption	What Happens
v1	noAuthNoPriv	Community String	No	Uses a community string match for authentication.
v2c	noAuthNoPriv	Community String	No	Uses a community string match for authentication.
v3	noAuthNoPriv	Username	No	Uses a username match for authentication.
v3	authNoPriv	Message Digest 5 (MD5) or Secure Hash Algorithm (SHA)	No	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms.
v3	authPriv	MD5 or SHA	Data Encryption Standard (DES)	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms. Provides DES 56-bit encryption in addition to authentication based on the CBC-DES (DES-56) standard.

Note SNMPv2p (SNMPv2 Classic) is not supported in Cisco IOS Release 11.2 and later releases. SNMPv2c replaces the Party-based Administrative and Security Framework of SNMPv2p with a Community-based Administrative Framework. SNMPv2c retained the bulk retrieval and error handling capabilities of SNMPv2p.

You must configure an SNMP agent to use the version of SNMP supported by the management station. An agent can communicate with multiple managers. You can configure the Cisco IOS software to support communications with one management station using the SNMPv1 protocol, one using the SNMPv2c protocol, and another using SNMPv3.

SNMPv3 supports RFCs 1901 to 1908, 2104, 2206, 2213, 2214, and 2271 to 2275. For additional information about SNMPv3, see RFC 2570, *Introduction to Version 3 of the Internet-standard Network Management Framework* (this is not a standards document).

Detailed Interface Registration Information

The Interface Index Display for SNMP feature introduces new commands and command modifications that allow advanced users of SNMP to view information about the interface registrations directly on the managed agent. You can display MIB information from the agent without using an external NMS.

Note For the purposes of this document, the agent is a routing device running Cisco software.

This feature addresses three objects in the Interfaces MIB: ifIndex, ifAlias, and ifName. For a complete definition of these objects, see the IF-MIB.my file available from the Cisco SNMPv2 MIB website.

Interface Index

The ifIndex object (ifEntry 1) is called the Interface Index. The Interface Index is a unique value greater than zero that identifies each interface or subinterface on the managed device. This value becomes the interface index identification number.

The CLI command **show snmp mib ifmib ifindex** allows you to view the SNMP Interface Index Identification numbers assigned to interfaces and subinterfaces. An NMS is not required.

Interface Alias

The ifAlias object (ifXEntry 18) is called the Interface Alias. The Interface Alias is a user-specified description of an interface used for SNMP network management. The ifAlias is an object in the Interfaces Group MIB (IF-MIB) that can be set by a network manager to “name” an interface. The ifAlias value for an interface or subinterface can be set using the **description** command in interface configuration mode or subinterface configuration mode or by using a Set operation from an NMS. Previously, ifAlias descriptions for subinterfaces were limited to 64 characters. (The OLD-CISCO-INTERFACES-MIB allows up to 255 characters for the locIfDescr MIB variable, but this MIB does not support subinterfaces.) A new CLI command, **snmp ifmib ifalias long**, configures the system to handle IfAlias descriptions of up to 256 characters. IfAlias descriptions appear in the output of the CLI **show interfaces** command.

Interface Name

The ifName object (ifXEntry 1) is the textual name of the interface. The purpose of the ifName object is to cross reference the CLI representation of a given interface. The value of this object is the name of the interface as assigned by the local device and is generally suitable for use in CLI commands. If there is no local name or this object is otherwise not applicable, this object contains a zero-length string. No commands introduced by this feature affect the ifName object, but it is discussed here to show its relation to the ifIndex and ifAlias objects.

The **show snmp mib** command shows all objects in the MIB on a Cisco device (similar to a mibwalk). The objects in the MIB tree are sorted using lexical ordering, meaning that object identifiers are sorted in sequential, numerical order. Lexical ordering is important when using the GetNext operation from an NMS because these operations take an object identifier (OID) or a partial OID as input and return the next object from the MIB tree based on the lexical ordering of the tree.

Note If an SNMP table query (SNMP MIB Walk) is performed on QOS MIB, you might see an increase in CPU utilization and this can occasionally lead to a session time out. As an alternative, use SNMP GET operation to retrieve a limited number of elements.

SNMP Support for VPNs

The SNMP Support for VPNs feature allows SNMP traps and informs to be sent and received using VPN routing and forwarding (VRF) tables. In particular, this feature adds support to the Cisco IOS software for sending and receiving SNMP traps and informs specific to individual VPNs.

A VPN is a network that provides high connectivity transfers on a shared system with the same usage guidelines as a private network. A VPN can be built on the Internet over IP, Frame Relay, or ATM networks.

A VRF stores per-VPN routing data. It defines the VPN membership of a customer site attached to the network access server (NAS). A VRF consists of an IP routing table, a derived Cisco Express Forwarding table, and guidelines and routing protocol parameters that control the information that is included in the routing table.

The SNMP Support for VPNs feature provides configuration commands that allow users to associate SNMP agents and managers with specific VRFs. The specified VRF is used for sending SNMP traps and informs and responses between agents and managers. If a VRF is not specified, the default routing table for the VPN is used.

Support for VPNs allows you to configure an SNMP agent to accept only SNMP requests from a certain set of VPNs. With this configuration, service providers can provide network management services to their customers, so customers can manage all user VPN devices.

Interface Index Persistence

One of the identifiers most commonly used in SNMP-based network management applications is the interface index (IfIndex) value. IfIndex is a unique identifying number associated with a physical or logical interface; as far as most software is concerned, the ifIndex is the name of the interface.

Although there is no requirement in the relevant RFCs that the correspondence between particular ifIndex values and their interfaces be maintained across reboots, applications such as device inventory, billing, and fault detection increasingly depend on the maintenance of this correspondence.

This feature adds support for an ifIndex value that can persist across reboots, allowing users to avoid the workarounds previously required for consistent interface identification.

It is currently possible to poll the device at regular intervals to correlate the interfaces to the ifIndex, but it is not practical to poll this interface constantly. If this data is not correlated constantly, however, the data may be made invalid because of a reboot or the insertion of a new card into the device in between polls. Therefore, ifIndex persistence is the only way to guarantee data integrity.

IfIndex persistence means that the mapping between the ifDescr object values and the ifIndex object values (generated from the IF-MIB) will be retained across reboots.

Benefits of Interface Index Persistence

Association of Interfaces with Traffic Targets for Network Management

The Interface Index Persistence feature allows for greater accuracy when collecting and processing network management data by uniquely identifying input and output interfaces for traffic flows and SNMP statistics. Relating each interface to a known entity (such as an ISP customer) allows network management data to be more effectively utilized.

Accuracy for Mediation, Fault Detection, and Billing

Network data is increasingly being used worldwide for usage-based billing, network planning, policy enforcement, and trend analysis. The ifIndex information is used to identify input and output interfaces for traffic flows and SNMP statistics. Inability to reliably relate each interface to a known entity, such as a customer, invalidates the data.

Event MIB

The Event MIB provides the ability to monitor MIB objects on a local or remote system using SNMP and initiate simple actions whenever a trigger condition is met; for example, an SNMP trap can be generated when an object is modified. When the notifications are triggered through events, the NMS does not need to constantly poll managed devices to track changes.

By allowing the SNMP notifications to take place only when a specified condition is met, the Event MIB reduces the load on affected devices and improves the scalability of network management solutions.

The Event MIB operates based on event, object lists configured for the event, event action, trigger, and trigger test.

Events

The event table defines the activities to be performed when an event is triggered. These activities include sending a notification and setting a MIB object. The event table has supplementary tables for additional objects that are configured according to event action. If the event action is set to notification, notifications are sent out whenever the object configured for that event is modified.

Object List

The object table lists objects that can be added to notifications based on trigger, trigger test type, or the event that sends a notification. The Event MIB allows wildcarding, which enables you to monitor multiple instances of an object. To specify a group of object identifiers, you can use the wildcard option.

Trigger

The trigger table defines conditions to trigger events. The trigger table lists the objects to be monitored and associates each trigger with an event. An event occurs when a trigger is activated. To create a trigger, you should configure a trigger entry in the mteTriggerTable of the Event MIB. This trigger entry specifies the object identifier of the object to be monitored. Each trigger is configured to monitor a single object or a group of objects specified by a wildcard (*). The Event MIB process checks the state of the monitored object at specified intervals.

Trigger Test

The trigger table has supplementary tables for additional objects that are configured based on the type of test performed for a trigger. For each trigger entry type such as existence, threshold, or Boolean, the corresponding tables (existence, threshold, and Boolean tables) are populated with the information required to perform the test. The Event MIB allows you to set event triggers based on existence, threshold, and Boolean trigger types. When the specified test on an object returns a value of *true*, the trigger is activated. You can configure the Event MIB to send out notifications to the interested host when a trigger is activated.

Expression MIB

The Expression MIB allows you to create expressions based on a combination of objects. The expressions are evaluated according to the sampling method. The Expression MIB supports the following types of object sampling:

- Absolute
- Delta
- Changed

If there are no delta or change values in an expression, the expression is evaluated when a requester attempts to read the value of expression. In this case, all requesters get a newly calculated value.

For expressions with delta or change values, evaluation is performed for every sampling. In this case, requesters get the value as of the last sample period.

Absolute Sampling

Absolute sampling uses the value of the MIB object during sampling.

Delta Sampling

Delta sampling is used for expressions with counters that are identified based on delta (difference) from one sample to the next. Delta sampling requires the application to do continuous sampling, because it uses the value of the last sample.

Changed Sampling

Changed sampling uses the changed value of the object since the last sample.

SNMP Notification Logging

Systems that support SNMP often need a mechanism for recording notification information. This mechanism protects against notifications being lost because they exceeded retransmission limits. The Notification Log MIB provides a common infrastructure for other MIBs in the form of a local logging function. The SNMP Notification Logging feature adds Cisco command line interface commands to change the size of the notification log, to set the global ageout value for the log, and to display logging summaries at the command line. The Notification Log MIB improves notification tracking and provides a central location for tracking all MIBs.

You can globally enable or disable authenticationFailure, linkUp, linkDown, warmStart, and coldStart traps or informs individually. (These traps constitute the “generic traps” defined in RFC 1157.) Note that linkUp

and linkDown notifications are enabled by default on specific interfaces but will not be sent unless they are enabled globally.

Note The Notification Log MIB supports notification logging on the default log only.

How to Configure SNMP Support

There is no specific command that you use to enable SNMP. The first **snmp-server** command that you enter enables the supported versions of SNMP. All other configurations are optional.

Configuring System Information

You can set the system contact, location, and serial number of the SNMP agent so that these descriptions can be accessed through the configuration file. Although the configuration steps described in this section are optional, configuring the basic information is recommended because it may be useful when troubleshooting your configuration. In addition, the first **snmp-server** command that you issue enables SNMP on the device.

Perform this task as needed.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server contact** *text*
4. **snmp-server location** *text*
5. **snmp-server chassis-id** *number*
6. end
7. **show snmp contact**
8. **show snmp location**
9. **show snmp chassis**

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	**snmp-server contact** *text* **Example:** `Device(config)# snmp-server contact NameOne`	Sets the system contact string.
Step 4	**snmp-server location** *text* **Example:** `Device(config)# snmp-server location LocationOne`	Sets the system location string.
Step 5	**snmp-server chassis-id** *number* **Example:** `Device(config)# snmp-server chassis-id 015A619T`	Sets the system serial number.
Step 6	end **Example:** `Device(config)# end`	Exits global configuration mode.
Step 7	**show snmp contact** **Example:** `Device# show snmp contact`	(Optional) Displays the contact strings configured for the system.
Step 8	**show snmp location** **Example:** `Device# show snmp location`	(Optional) Displays the location string configured for the system.
Step 9	**show snmp chassis** **Example:** `Device# show snmp chassis`	(Optional) Displays the system serial number.

Configuring SNMP Versions 1 and 2

When you configure SNMP versions 1 and 2, you can optionally create or modify views for community strings to limit which MIB objects an SNMP manager can access.

Perform the following tasks when configuring SNMP version 1 or version 2.

Prerequisites

- An established SNMP community string that defines the relationship between the SNMP manager and the agent.

- A host defined to be the recipient of SNMP notifications.
- Use **no snmp-server** command to turn off the SNMP services, such as listening UDP ports and processes. To remove the individual SNMP configs, use no form of the respective SNMP config commands.

Creating or Modifying an SNMP View Record

You can assign views to community strings to limit which MIB objects an SNMP manager can access. You can use a predefined view or create your own view. If you are using a predefined view or no view at all, skip this task.

Perform this task to create or modify an SNMP view record.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server view** *view-name oid-tree* {**included** | **excluded**}
4. **no snmp-server view** *view-name oid-tree* {**included** | **excluded**}
5. **end**
6. show snmp view

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	**snmp-server view** *view-name oid-tree* {**included**	**excluded**} **Example:** `Device(config)# snmp-server view mib2 mib-2 included`	Creates a view record. • In this example, the mib2 view that includes all objects in the MIB-II subtree is created. **Note** You can use this command multiple times to create the same view record. If a view record for the same OID value is created multiple times, the latest entry of the object identifier takes precedence.
Step 4	**no snmp-server view** *view-name oid-tree* {**included**	**excluded**} **Example:**	Removes a server view.

	Command or Action	Purpose
	`Device(config)# no snmp-server view mib2 mib-2 included`	
Step 5	**end** **Example:** `Device(config)# end`	Exits global configuration mode.
Step 6	show snmp view **Example:** `Device# show snmp view`	(Optional) Displays a view of the MIBs associated with SNMP.

Creating or Modifying Access Control for an SNMP Community

Use an SNMP community string to define the relationship between the SNMP manager and the agent. The community string acts like a password to regulate access to the agent on the device. Optionally, you can specify one or more of the following characteristics associated with the string:

- An access list of IP addresses of the SNMP managers that are permitted to use the community string to gain access to the agent.
- A MIB view, which defines the subset of all MIB objects accessible to the given community.
- Read and write or read-only permission for the MIB objects accessible to the community.

Perform this task to create or modify a community string.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**] [**ipv6** *nacl*] [*access-list-number*]
4. **no snmp-server community** *string*
5. **end**
6. **show snmp community**

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:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**] [**ipv6** *nacl*] [*access-list-number*] **Example:** `Device(config)# snmp-server community comaccess ro 4`	Defines the community access string. • You can configure one or more community strings.
Step 4	**no snmp-server community** *string* **Example:** `Device(config)# no snmp-server community comaccess`	Removes the community string from the configuration.
Step 5	**end** **Example:** `Device(config)# end`	Exits global configuration mode.
Step 6	**show snmp community** **Example:** `Device# show snmp community`	(Optional) Displays the community access strings configured for the system.

Configuring a Recipient of an SNMP Trap Operation

SNMP traps are unreliable because the receiver does not send acknowledgments when it receives traps. The sender does not know if the traps were received. However, an SNMP entity that receives an inform acknowledges the message with an SNMP response PDU. If the sender never receives the response, the inform can be sent again. Thus, informs are more likely to reach their intended destination.

Compared to traps, informs consume more resources in the agent and in the network. Unlike a trap, which is discarded as soon as it is sent, an inform must be held in memory until a response is received or the request times out. Also, traps are sent only once; an inform may be sent several times. The retries increase traffic and overhead on the network.

If you do not enter a **snmp-server host** command, no notifications are sent. To configure the device to send SNMP notifications, you must enter at least one **snmp-server host** command. If you enter the command without keywords, all trap types are enabled for the host.

To enable multiple hosts, you must issue a separate **snmp-server host** command for each host. You can specify multiple notification types in the command for each host.

When multiple **snmp-server host** commands are given for the same host and type of notification, each succeeding command overwrites the previous command. Only the last **snmp-server host** command will be in effect. For example, if you enter an **snmp-server host inform** command for a host and then enter another **snmp-server host inform** command for the same host, the second command replaces the first.

The **snmp-server host** command is used in conjunction with the **snmp-server enable** command. Use the **snmp-server enable** command to specify which SNMP notifications are sent globally. For a host to receive

most notifications, at least one **snmp-server enable** command and the **snmp-server host** command for that host must be enabled.

Some notification types cannot be controlled with the **snmp-server enable** command. For example, some notification types are always enabled and others are enabled by a different command. For example, the linkUpDown notifications are controlled by the **snmp trap link-status** command. These notification types do not require an **snmp-server enable** command.

A *notification-type* option's availability depends on the device type and the Cisco IOS software features supported on the device. For example, the envmon notification type is available only if the environmental monitor is part of the system. To see what notification types are available on your system, use the command help (?) at the end of the **snmp-server host** command.

Perform this task to configure the recipient of an SNMP trap operation.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server host** *host-id* [**traps** | **informs**] [**version** {**1**| **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port-number*] [*notification-type*]
4. **exit**
5. **show snmp host**

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	**snmp-server host** *host-id* [**traps** | **informs**] [**version** {**1**| **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port-number*] [*notification-type*] **Example:** `Device(config)# snmp-server host 172.16.1.27 informs version 2c public alarms`	Specifies whether you want the SNMP notifications sent as traps or informs, the version of SNMP to use, the security level of the notifications (for SNMPv3), and the recipient (host) of the notifications.
Step 4	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode.
Step 5	**show snmp host** **Example:** `Device# show snmp host`	(Optional) Displays the SNMP notifications sent as traps, the version of SNMP, and the host IP address of the notifications.

Examples

The following example shows the host information configured for SNMP notifications:

```
Device> enable
Device# configure terminal
Device(config)# snmp-server host 10.2.28.1 informs version 2c public
Device(config)# exit
Device# show snmp host

Notification host: 10.2.28.1 udp-port: 162   type: inform
user: public    security model: v2c
traps: 00001000.00000000.00000000
```

Configuring SNMP Version 3

When you configure SNMPv3 and you want to use the SNMPv3 security mechanism for handling SNMP packets, you must establish SNMP groups and users with passwords.

Perform the following tasks to configure SNMPv3.

Specifying SNMP-Server Group Names

SNMPv3 is a security model. A security model is an authentication strategy that is set up for a user and the group in which the user resides.

No default values exist for authentication or privacy algorithms when you configure the **snmp-server group** command. Also, no default passwords exist. For information about specifying a MD5 password, see the documentation for the **snmp-server user** command.

Perform this task to specify a new SNMP group or a table that maps SNMP users to SNMP views.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server group** [*groupname* {**v1** | **v2c** | **v3** [**auth** | **noauth** | **priv**]}] [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*]
4. **exit**
5. **show snmp group**

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:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**snmp-server group** [*groupname* {**v1** | **v2c** | **v3** [**auth** | **noauth** | **priv**]}] [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*] **Example:** `Device(config)# snmp-server group group1 v3 auth access lmnop`	Configures the SNMP server group to enable authentication for members of a specified named access list. • In this example, the SNMP server group *group1* isconfigured to enable user authentication for members of the named access list *lmnop*.
Step 4	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode.
Step 5	**show snmp group** **Example:** `Device# show snmp group`	Displays information about each SNMP group on the network.

Examples

The following example shows information about each SNMP group on the network:

```
Device# show snmp group
groupname: ILMI                              security model:v1
readview : *ilmi                             writeview: *ilmi
notifyview: <no notifyview specified>
row status: active
groupname: ILMI                              security model:v2c
readview : *ilmi                             writeview: *ilmi
notifyview: <no notifyview specified>
row status: active
groupname: group1                            security model:v3 auth
readview : v1default                         writeview: <no writeview specified>
notifyview: <no notifyview specified>
row status: active                           access-list:lmnop
groupname: public                            security model:v1
readview : <no readview specified>           writeview: <no writeview specified>
notifyview: <no notifyview specified>
row status: active
```

Configuring SNMP Server Users

To configure a remote user, specify the IP address or port number for the remote SNMP agent of the device where the user resides. Also, before you configure remote users for a particular agent, configure the SNMP engine ID, using the **snmp-server engineID** command with the remote option. The remote agent's SNMP engine ID is required when computing the authentication and privacy digests from the password. If the remote engine ID is not configured first, the configuration command will fail.

For the *privpassword* and *auth-password* arguments, the minimum length is one character; the recommended length is at least eight characters, and should include both letters and numbers.

SNMP passwords are localized using the SNMP engine ID of the authoritative SNMP engine. For informs, the authoritative SNMP agent is the remote agent. You must configure the remote agent's SNMP engine ID in the SNMP database before you can send proxy requests or informs to it.

Note Changing the engine ID after configuring the SNMP user does not allow the removal of the user. To remove the configurations, you need to first reconfigure all the SNMP configurations.

No default values exist for authentication or privacy algorithms when you configure the command. Also, no default passwords exist. The minimum length for a password is one character, although we recommend using at least eight characters for security. If you forget a password, you cannot recover it and will need to reconfigure the user. You can specify either a plain text password or a localized MD5 digest.

If you have the localized MD5 or SHA digest, you can specify that string instead of the plain text password. The digest should be formatted as aa:bb:cc:dd where aa, bb, and cc are hexadecimal values. Also, the digest should be exactly 16 octets in length.

Perform this task to add a new user to an SNMP group.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server engineID** {**local** *engine-id* | **remote** *ip-address* [**udp-port** *udp-port-number*] [**vrf** *vrf-name*] *engine-id-string*}
4. **snmp-server user** *username groupname* [**remote** *ip-address* [**udp-port** *port*]] {**v1** | **v2c** | **v3** [**encrypted**] [**auth** {**md5** | **sha**} *auth-password*]} [**access** *access-list*]
5. **exit**
6. **show snmp user** [*username*]
7. **show snmp engineID**

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	**snmp-server engineID** {**local** *engine-id* | **remote** *ip-address* [**udp-port** *udp-port-number*] [**vrf** *vrf-name*] *engine-id-string*}	Configures the SNMP engine ID. • In this example, the SNMP engine ID is configured for a remote user.

	Command or Action	Purpose
	Example: `Device(config)# snmp-server engineID remote 172.12.15.4 udp-port 120 1a2833c0129a`	
Step 4	**snmp-server user** *username groupname* [**remote** *ip-address* [**udp-port** *port*]] {**v1** | **v2c** | **v3** [**encrypted**] [**auth** {**md5** | **sha**} *auth-password*]} [**access** *access-list*] **Example:** `Device(config)# snmp-server user user1 group1 v3 auth md5 password123`	Configures a new user to an SNMP group with the plain text password "password123" for the user "user1" in the SNMPv3 group "group1".
Step 5	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 6	**show snmp user** [*username*] **Example:** `Device# show snmp user user1`	Displays the information about the configured characteristics of an SNMP user.
Step 7	**show snmp engineID** **Example:** `Device# show snmp engineID`	(Optional) Displays information about the SNMP engine ID configured for an SNMP user.

Examples

The following example shows the information about the configured characteristics of the SNMP user1:

```
Device# show snmp user user1
User name: user1
Engine ID: 000000090200000000C025808
storage-type: nonvolatile       active access-list: 10
Rowstatus: active
Authentication Protocol: MD5
Privacy protocol: None
Group name: group1
```

Configuring a Device as an SNMP Manager

Perform this task to enable the SNMP manager process and to set the session timeout value.

SUMMARY STEPS

1. **enable**
2. **configure terminal**

3. **snmp-server manager**
4. **snmp-server manager session-timeout** *seconds*
5. **end**
6. **show snmp**
7. **show snmp sessions [brief]**
8. **show snmp pending**

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	**snmp-server manager** **Example:** `Device(config)# snmp-server manager`	Enables the SNMP manager.
Step 4	**snmp-server manager session-timeout** *seconds* **Example:** `Device(config)# snmp-server manager session-timeout 30`	(Optional) Changes the session timeout value.
Step 5	**end** **Example:** `Device(config)# end`	Exits global configuration mode.
Step 6	**show snmp** **Example:** `Device# show snmp`	(Optional) Displays the status of SNMP communications.
Step 7	**show snmp sessions [brief]** **Example:** `Device# show snmp sessions`	(Optional) Displays the status of SNMP sessions.
Step 8	**show snmp pending** **Example:** `Device# show snmp pending`	(Optional) Displays the current set of pending SNMP requests.

Examples

The following example shows the status of SNMP communications:

```
Device# show snmp

Chassis: 01506199
37 SNMP packets input
    0 Bad SNMP version errors
    4 Unknown community name
    0 Illegal operation for community name supplied
    0 Encoding errors
    24 Number of requested variables
    0 Number of altered variables
    0 Get-request PDUs
    28 Get-next PDUs
    0 Set-request PDUs
78 SNMP packets output
    0 Too big errors (Maximum packet size 1500)
    0 No such name errors
    0 Bad values errors
    0 General errors
    24 Response PDUs
    13 Trap PDUs
SNMP logging: enabled
    Logging to 172.17.58.33.162, 0/10, 13 sent, 0 dropped.
SNMP Manager-role output packets
    4 Get-request PDUs
    4 Get-next PDUs
    6 Get-bulk PDUs
    4 Set-request PDUs
    23 Inform-request PDUs
    30 Timeouts
    0 Drops
SNMP Manager-role input packets
    0 Inform response PDUs
    2 Trap PDUs
    7 Response PDUs
    1 Responses with errors
SNMP informs: enabled
    Informs in flight 0/25 (current/max)
    Logging to 172.17.217.141.162
        4 sent, 0 in-flight, 1 retries, 0 failed, 0 dropped
    Logging to 172.17.58.33.162
        0 sent, 0 in-flight, 0 retries, 0 failed, 0 dropped
```

The following example displays the status of SNMP sessions:

```
Device# show snmp sessions

Destination: 172.17.58.33.162, V2C community: public
  Round-trip-times: 0/0/0 (min/max/last)
  packets output
    0 Gets, 0 GetNexts, 0 GetBulks, 0 Sets, 4 Informs
    0 Timeouts, 0 Drops
  packets input
    0 Traps, 0 Informs, 0 Responses (0 errors)
Destination: 172.17.217.141.162, V2C community: public, Expires in 575 secs
  Round-trip-times: 1/1/1 (min/max/last)
  packets output
    0 Gets, 0 GetNexts, 0 GetBulks, 0 Sets, 4 Informs
    0 Timeouts, 0 Drops
```

```
 packets input
    0 Traps, 0 Informs, 4 Responses (0 errors)
```

The following example shows the current set of pending SNMP requests:

```
Device# show snmp pending

req id: 47, dest: 172.17.58.33.161, V2C community: public, Expires in 5 secs
req id: 49, dest: 172.17.58.33.161, V2C community: public, Expires in 6 secs
req id: 51, dest: 172.17.58.33.161, V2C community: public, Expires in 6 secs
req id: 53, dest: 172.17.58.33.161, V2C community: public, Expires in 8 secs
```

Enabling the SNMP Manager

Perform this task to enable the SNMP manager process and to set the session timeout value.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server manager**
4. **snmp-server manager session-timeout** *seconds*
5. **exit**
6. **show snmp**
7. **show snmp sessions [brief]**
8. **show snmp pending**

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	**snmp-server manager** **Example:** `Device(config)# snmp-server manager`	Enables the SNMP manager.
Step 4	**snmp-server manager session-timeout** *seconds* **Example:** `Device(config)# snmp-server manager session-timeout 30`	(Optional) Changes the session timeout value.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode.
Step 6	**show snmp** **Example:** `Device# show snmp`	(Optional) Displays the status of SNMP communications.
Step 7	**show snmp sessions [brief]** **Example:** `Device# show snmp sessions`	(Optional) Displays displays the status of SNMP sessions.
Step 8	**show snmp pending** **Example:** `Device# show snmp pending`	(Optional) Displays the current set of pending SNMP requests.

Examples

The following example shows the status of SNMP communications:

```
Device# show snmp
Chassis: 01506199
37 SNMP packets input
    0 Bad SNMP version errors
    4 Unknown community name
    0 Illegal operation for community name supplied
    0 Encoding errors
    24 Number of requested variables
    0 Number of altered variables
    0 Get-request PDUs
    28 Get-next PDUs
    0 Set-request PDUs
78 SNMP packets output
    0 Too big errors (Maximum packet size 1500)
    0 No such name errors
    0 Bad values errors
    0 General errors
    24 Response PDUs
    13 Trap PDUs
SNMP logging: enabled
    Logging to 172.17.58.33.162, 0/10, 13 sent, 0 dropped.
SNMP Manager-role output packets
    4 Get-request PDUs
    4 Get-next PDUs
    6 Get-bulk PDUs
    4 Set-request PDUs
    23 Inform-request PDUs
    30 Timeouts
    0 Drops
```

```
SNMP Manager-role input packets
    0 Inform response PDUs
    2 Trap PDUs
    7 Response PDUs
    1 Responses with errors
SNMP informs: enabled
    Informs in flight 0/25 (current/max)
    Logging to 172.17.217.141.162
        4 sent, 0 in-flight, 1 retries, 0 failed, 0 dropped
    Logging to 172.17.58.33.162
        0 sent, 0 in-flight, 0 retries, 0 failed, 0 dropped
```

The following example displays the status of SNMP sessions:

```
Device# show snmp sessions
Destination: 172.17.58.33.162, V2C community: public
  Round-trip-times: 0/0/0 (min/max/last)
  packets output
    0 Gets, 0 GetNexts, 0 GetBulks, 0 Sets, 4 Informs
    0 Timeouts, 0 Drops
  packets input
    0 Traps, 0 Informs, 0 Responses (0 errors)
Destination: 172.17.217.141.162, V2C community: public, Expires in 575 secs
  Round-trip-times: 1/1/1 (min/max/last)
  packets output
    0 Gets, 0 GetNexts, 0 GetBulks, 0 Sets, 4 Informs
    0 Timeouts, 0 Drops
 packets input
    0 Traps, 0 Informs, 4 Responses (0 errors)
```

The following example shows the current set of pending SNMP requests:

```
Device# show snmp pending
req id: 47, dest: 172.17.58.33.161, V2C community: public, Expires in 5 secs
req id: 49, dest: 172.17.58.33.161, V2C community: public, Expires in 6 secs
req id: 51, dest: 172.17.58.33.161, V2C community: public, Expires in 6 secs
req id: 53, dest: 172.17.58.33.161, V2C community: public, Expires in 8 secs
```

Enabling the SNMP Agent Shutdown Mechanism

Using SNMP packets, a network management tool can send messages to users on virtual terminals and on the console. This facility operates in a similar fashion to the **send** EXEC command; however, the SNMP request that causes the message to be issued to the users also specifies the action to be taken after the message is delivered. One possible action is a shutdown request. After a system is shut down, typically it is reloaded. Because the ability to cause a reload from the network is a powerful feature, it is protected by the **snmp-server system-shutdown** global configuration command. If you do not issue this command, the shutdown mechanism is not enabled.

Perform this task to enable the SNMP agent shutdown mechanism.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server system-shutdown**
4. end

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	**snmp-server system-shutdown** **Example:** `Device(config)# snmp-server system-shutdown`	Enables system shutdown using the SNMP message reload feature.
Step 4	end **Example:** `Device(config)# end`	Exits global configuration mode.

Defining the Maximum SNMP Agent Packet Size

You can define the maximum packet size permitted when the SNMP agent is receiving a request or generating a reply.

Perform this task to set the maximum permitted packet size.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server packetsize** *byte-count*
4. **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:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**snmp-server packetsize** *byte-count* **Example:** `Device(config)# snmp-server packetsize 512`	Establishes the maximum packet size.
Step 4	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Limiting the Number of TFTP Servers Used via SNMP

You can limit the number of TFTP servers used for saving and loading configuration files via SNMP by using an access list. Limiting the use of TFTP servers in this way conserves system resources and centralizes the operation for manageability.

Perform this task to limit the number of TFTP servers.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server tftp-server-list** *number*
4. **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	**snmp-server tftp-server-list** *number* **Example:** `Device(config)# snmp-server tftp-server-list 12`	Limits the number of TFTP servers used for configuration file copies via SNMP to the servers in an access list.
Step 4	**exit** **Example:**	Exits global configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config)# exit`	

Troubleshooting Tips

To monitor SNMP trap activity in real time for the purposes of troubleshooting, use the SNMP **debug** commands, including the **debug snmp packet** EXEC command. For documentation of SNMP **debug** commands, see the *Cisco IOS Debug Command Reference.*

Disabling the SNMP Agent

Perform this task to disable any version of an SNMP agent.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **no snmp-server**
4. **end**

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 snmp-server** **Example:** `Device(config)# no snmp-server`	Disables SNMP agent operation.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode.

Configuring SNMP Notifications

To configure a device to send SNMP traps or informs, perform the tasks described in the following sections:

Note Many snmp-server commands use the keyword **traps** in their command syntax. Unless there is an option within the command to specify either traps or informs, the keyword **traps** should be taken to mean traps, informs, or both. Use the **snmp-server host** command to specify whether you want SNMP notifications to be sent as traps or informs. To use informs, the SNMP manager (also known as the SNMP proxy manager) must be available and enabled on a device. Earlier, the SNMP manager was available only with Cisco IOS PLUS images. However, the SNMP manager is now available with all Cisco software releases that support SNMP. Use Cisco Feature Navigator for information about SNMP manager support for Cisco software releases. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.

Note If no notify view is defined, no objects can send notifications to members of the group. The possible ways to configure notify view include to configure it manually while defining the snm-server group and automatically, when a user in the group is bound to a notification host target.

Configuring the Device to Send SNMP Notifications

Perform this task to configure the device to send traps or informs to a host.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server engineID remote** *remote-ip-address remote-engineID*
4. **snmp-server user** *username groupname* [**remote** *host* [**udp-port** *port*] {**v1** | **v2c** | **v3** [**encrypted**] [**auth** {**md5** | **sha**} *auth-password*]} [**access** *access-list*]
5. **snmp-server group** *groupname* {**v1** | **v2c** | **v3** {**auth** | **noauth** | **priv**}} [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*]
6. **snmp-server host** *host* [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [*notification-type*]
7. **snmp-server enable traps** [*notification-type* [*notification-options*]]
8. end

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	**snmp-server engineID remote** *remote-ip-address remote-engineID* **Example:** `Device(config)# snmp-server engineID remote 172.16.20.3 80000009030000B064EFE100`	Specifies the SNMP engine ID and configures the VRF name traps-vrf for SNMP communications with the remote device at 172.16.20.3.
Step 4	**snmp-server user** *username groupname* [**remote** *host* [**udp-port** *port*] {**v1** | **v2c** | **v3** [**encrypted**] [**auth** {**md5** | **sha**} *auth-password*]} [**access** *access-list*] **Example:** `Device(config)# snmp-server user abcd public v3 encrypted auth md5 cisco123`	Configures a local or remote user to an SNMP group. **Note** You cannot configure a remote user for an address without first configuring the engine ID for that remote host. This restriction is imposed in the design of these commands; if you try to configure the user before the host, you will receive a warning message and the command will not be executed. Use the **snmp-server engineid remote** command to specify the engine ID for a remote host.
Step 5	**snmp-server group** *groupname* {**v1** | **v2c** | **v3** {**auth** | **noauth** | **priv**}} [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*] **Example:** `Device(config)# snmp-server group GROUP1 v2c auth read viewA write viewA notify viewB`	Configures an SNMP group.
Step 6	**snmp-server host** *host* [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [*notification-type*] **Example:** `Device(config)# snmp-server host example.com informs version 3 public`	Specifies whether you want the SNMP notifications sent as traps or informs, the version of SNMP to use, the security level of the notifications (for SNMPv3), and the recipient (host) of the notifications. • The **snmp-server host** command specifies which hosts will receive SNMP notifications, and whether you want the notifications sent as traps or informs.
Step 7	**snmp-server enable traps** [*notification-type* [*notification-options*]] **Example:** `Device(config)# snmp-server enable traps bgp`	Enables sending of traps or informs and specifies the type of notifications to be sent. • If a *notification-type* is not specified, all supported notification are enabled on the device. • To discover which notifications are available on your device, enter the **snmp-server enable traps ?** command. • The **snmp-server enable traps** command globally enables the production mechanism for the specified notification types (such as Border Gateway Protocol [BGP] traps, config traps, entity traps, Hot Standby Device Protocol [HSDP] traps, and so on).

	Command or Action	Purpose
Step 8	end **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Enabling Syslog Trap Messages

You can enable Syslog traps using the **snmp-server enable traps syslog** command.

After you enable Syslog traps, you have to specify the trap message severity. Use the **logging snmp-trap** command to specify the trap level. By default, the command enables severity 0 to 4. If you want to enable all the severities, use the following form of the command:

logging snmp-trap 0 7

You can also enable individual trap levels using the following forms of the command:

logging snmp-trap emergencies: Enables only severity 0 traps.

logging snmp-trap alert: Enables only severity 1 traps.

Similarly, you can separately configure other trap levels.

Note that, along with the above configuration, Syslog history command also needs to be applied. Without this configuration, Syslog traps are not sent.

Use the following command to enable the Syslog history command:

logging history informational: Enables traps up to informational level which is severity 6.

Changing Notification Operation Values

You can specify a value other than the default for the source interface, message (packet) queue length for each host, or retransmission interval.

Perform this task to change notification operation values as needed.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server trap-source** *interface*
4. **snmp-server queue-length** *length*
5. **snmp-server trap-timeout** *seconds*
6. **snmp-server informs** [**retries** *retries*] [**timeout** *seconds*] [**pending** *pending*]

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	**snmp-server trap-source** *interface* **Example:** `Device(config)# snmp-server trap-source FastEthernet 2/1`	Sets the IP address for the Fast Ethernet interface in slot2, port 1 as the source for all SNMP notifications.
Step 4	**snmp-server queue-length** *length* **Example:** `Device(config)# snmp-server queue-length 50`	Establishes the message queue length for each notification. • This example shows the queue length set to 50 entries.
Step 5	**snmp-server trap-timeout** *seconds* **Example:** `Device(config)# snmp-server trap-timeout 30`	Defines how often to resend notifications on the retransmission queue.
Step 6	**snmp-server informs** [**retries** *retries*] [**timeout** *seconds*] [**pending** *pending*] **Example:** `Device(config)# snmp-server informs retries 10 timeout 30 pending 100`	Configures inform-specific operation values. • This example sets the maximum number of times to resend an inform, the number of seconds to wait for an acknowledgment before resending, and the maximum number of informs waiting for acknowledgments at any one time.

Controlling Individual RFC 1157 SNMP Traps

Perform this task to enable the authenticationFailure, linkUp, linkDown, warmStart, and coldStart notification types.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server enable traps snmp** [**authentication**] [**linkup**] [**linkdown**] [**warmstart**] [**coldstart**]
4. **interface** *type slot*/*port*
5. **no snmp-server link-status**
6. **end**
7. **end**
8. **show snmp mib ifmibtraps**

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	**snmp-server enable traps snmp** [**authentication**] [**linkup**] [**linkdown**] [**warmstart**] [**coldstart**] **Example:** `Device(config)# snmp-server enable traps snmp`	Enables RFC 1157 generic traps. • When used without any of the optional keywords, enables authenticationFailure, linkUp, linkDown, warmStart, and coldStart traps. • When used with keywords, enables only the trap types specified. For example, to globally enable only linkUp and linkDown SNMP traps or informs for all interfaces, use the **snmp-server enable traps snmp linkup linkdown** form of this command.
Step 4	**interface** *type slot/port* **Example:** `Device(config)# interface FastEthernet 0/0`	Enters interface configuration mode for a specific interface. **Note** To enable SNMP traps for individual interfaces such as Dialer, use the **snmp trap link-status permit duplicates** command in interface configuration mode. For example, to enter dialer interface configuration mode, enter the interface type as dialer.
Step 5	**no snmp-server link-status** **Example:** `Device(config-if)# no snmp-server link-status`	Disables the sending of linkUp and linkDown notifications for all generic interfaces.
Step 6	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode.
Step 7	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.
Step 8	**show snmp mib ifmibtraps**	

	Command or Action	Purpose
	Example: `Device# show snmp mib ifmib traps`	

Examples

The following example shows the status of linkup and linkdown traps for all interfaces configured for the system:

```
Device# show snmp mib ifmib traps

ifDescr   ifindex  TrapStatus
--------------------------------
FastEthernet  3/6 14  enabled
FastEthernet  3/19 27  enabled
GigabitEthernet  5/1 57  enabled
unrouted VLAN  1005 73  disabled
FastEthernet  3/4 12  enabled
FastEthernet  3/39 47  enabled
FastEthernet  3/28 36  enabled
FastEthernet  3/48 56  enabled
unrouted VLAN  1003 74  disabled
FastEthernet  3/2 10  enabled
Tunnel   0 66  enabled
SPAN RP Interface  64  disabled
Tunnel   10 67  enabled
FastEthernet  3/44 52  enabled
GigabitEthernet  1/3 3  enabled
FastEthernet  3/11 19  enabled
FastEthernet  3/46 54  enabled
GigabitEthernet  1/1 1  enabled
FastEthernet  3/13 21  enabled
unrouted VLAN  1 70  disabled
GigabitEthernet  1/4 4  enabled
FastEthernet  3/9 17  enabled
FastEthernet  3/16 24  enabled
FastEthernet  3/43 51  enabled
```

Configuring SNMP Notification Log Options

Perform this task to configure SNMP notification log options. These options allow you to control the log size and timing values. The SNMP log can become very large and long, if left unmodified.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib notification-log default**
4. **snmp mib notification-log globalageout** *seconds*
5. **snmp mib notification-log globalsize** *size*
6. **end**
7. **show snmp mib notification-log**

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	**snmp mib notification-log default** **Example:** `Device(config)# snmp mib notification-log default`	Creates an unnamed SNMP notification log.
Step 4	**snmp mib notification-log globalageout** *seconds* **Example:** `Device(config)# snmp mib notification-log globalageout 20`	Sets the maximum amount of time for which the SNMP notification log entries remain in the system memory. • In this example, the system is configured to delete entries in the SNMP notification log that were logged more than 20 minutes ago.
Step 5	**snmp mib notification-log globalsize** *size* **Example:** `Device(config)# snmp mib notification-log globalsize 600`	Sets the maximum number of entries that can be stored in all SNMP notification logs.
Step 6	**end** **Example:** `Device(config)# end`	Exits global configuration mode.
Step 7	**show snmp mib notification-log** **Example:** `Device# show snmp mib notification-log`	Displays information about the state of the local SNMP notification logging.

Examples

This example shows information about the state of local SNMP notification logging:

```
Device# show snmp mib notification-log

GlobalAgeout 20, GlobalEntryLimit 600
Total Notifications logged in all logs 0
Log Name"", Log entry Limit 600, Notifications logged 0
Logging status enabled
Created by cli
```

Configuring Interface Index Display and Interface Indexes and Long Name Support

The display of Interface Indexes lets advanced users of SNMP view information about the interface registrations directly on a managed agent. An external NMS is not required.

Configuration of Long Alias Names for the interfaces lets users configure the ifAlias (the object defined in the MIB whose length is restricted to 64) up to 255 bytes.

Before you begin

SNMP must be enabled on your system.

The Interface Index Display and Interface Alias Long Name Support feature is not supported on all Cisco platforms. Use Cisco Feature Navigator to find information about platform support and software image support.

Perform this task to configure the IF-MIB to retain ifAlias values of longer than 64 characters and to configure the ifAlias values for an interface.

Note To verify if the ifAlias description is longer than 64 characters, perform an SNMP MIB walk for the ifMIB ifAlias variable from an NMS and verify that the entire description is displayed in the values for ifXEntry.18.

The description for interfaces also appears in the output from the **more system:running config** privileged EXEC mode command.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp ifmib ifalias long**
4. **interface** *type number*
5. **description** *text-string*
6. **end**
7. **show snmp mib**
8. **show snmp mib ifmib ifindex** [*type number*] [**detail**] [**free-list**]

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	**snmp ifmib ifalias long** **Example:** `Device(config)# snmp ifmib ifalias long`	Configures the Interfaces MIB (IF-MIB) on the system to return ifAlias values of longer than 64 characters to a Network Management System. • If the ifAlias values are not configured using the **snmp ifmib ifalias long** command, the ifAlias description will be restricted to 64 characters.
Step 4	**interface** *type number* **Example:** `Device(config)# interface ethernet 2/4`	Enters interface configuration mode. • The form of this command varies depending on the interface being configured.
Step 5	**description** *text-string* **Example:** `Device(config)# description This text string description can be up to 256 characters long`	Configures a free-text description of the specified interface. • This description can be up to 240 characters in length and is stored as the ifAlias object value in the IF-MIB. • If the ifAlias values are not configured using the **snmp ifmib ifalias long** command, the ifAlias description for SNMP set and get operations is restricted to 64 characters, although the interface description is configured for more than 64 characters by using the **description** command.
Step 6	**end** **Example:** `Device(config)# end`	Exits global configuration mode.
Step 7	**show snmp mib** **Example:** `Device# show snmp mib`	Displays a list of MIB module instance identifiers registered on your system. • The resulting display could be lengthy.
Step 8	**show snmp mib ifmib ifindex** [*type number*] [**detail**] [**free-list**] **Example:** `Device# show snmp mib ifmib ifindex Ethernet 2/0`	Displays the Interfaces MIB ifIndex values registered on your system for all interfaces or the specified interface.

Examples

The following example lists the MIB module instance identifiers registered on your system. The resulting display could be lengthy. Only a small portion is shown here.

```
Device# show snmp mib
system.1
system.2
sysUpTime
system.4
```

```
system.5
system.6
system.7
system.8
sysOREntry.2
sysOREntry.3
sysOREntry.4
interfaces.1
ifEntry.1
ifEntry.2
ifEntry.3
ifEntry.4
ifEntry.5
ifEntry.6
ifEntry.7
ifEntry.8
ifEntry.9
ifEntry.10
ifEntry.11
 --More--
captureBufferEntry.2
captureBufferEntry.3
captureBufferEntry.4
captureBufferEntry.5
captureBufferEntry.6
captureBufferEntry.7
capture.3.1.1
eventEntry.1
eventEntry.2
eventEntry.3
eventEntry.4
eventEntry.5
eventEntry.6
eventEntry.7
logEntry.1
logEntry.2
logEntry.3
logEntry.4
rmon.10.1.1.2
rmon.10.1.1.3
rmon.10.1.1.4
rmon.10.1.1.5
rmon.10.1.1.6
rmon.10.1.1.7
rmon.10.2.1.2
rmon.10.2.1.3
rmon.10.3.1.2
```

The following example shows output for the Interfaces MIB ifIndex values registered on a system for a specific interface:

```
Device# show snmp mib ifmib ifindex Ethernet 2/0
Ethernet2/0: Ifindex = 2
```

The following example shows output for the Interfaces MIB ifIndex values registered on a system for all interfaces:

```
Device# show snmp mib ifmib ifindex
ATM1/0: Ifindex = 1
ATM1/0-aal5 layer: Ifindex = 12
ATM1/0-atm layer: Ifindex = 10
ATM1/0.0-aal5 layer: Ifindex = 13
ATM1/0.0-atm subif: Ifindex = 11
ATM1/0.9-aal5 layer: Ifindex = 32
ATM1/0.9-atm subif: Ifindex = 31
```

```
ATM1/0.99-aal5 layer: Ifindex = 36
ATM1/0.99-atm subif: Ifindex = 35
Ethernet2/0: Ifindex = 2
Ethernet2/1: Ifindex = 3
Ethernet2/2: Ifindex = 4
Ethernet2/3: Ifindex = 5
Null0: Ifindex = 14
Serial3/0: Ifindex = 6
Serial3/1: Ifindex = 7
Serial3/2: Ifindex = 8
Serial3/3: Ifindex = 9
```

Configuring Interface Index Persistence

The following sections contain the tasks to configure Interface Index Persistence:

Enabling and Disabling IfIndex Persistence Globally

Perform this task to enable IfIndex persistence globally.

Before you begin

The configuration tasks described in this section assume that you have configured SNMP on your routing device and are using SNMP to monitor network activity using the Cisco command line interface and/or an NMS application.

Note To save the **snmp-server ifindex persist** command, enable the **snmp service** using any of the **snmp server**config commands, except the **snmp-server ifindex persist** command.

The interface-specific ifIndex persistence command (**snmp ifindex persistence**) cannot be used on subinterfaces. A command applied to an interface is automatically applied to all subinterfaces associated with that interface.

Testing indicates that approximately 25 bytes of NVRAM storage are used by this feature per interface. There may be some boot delay exhibited on platforms with lower CPU speeds.

Note After ifIndex persistence commands have been entered, the configuration must be saved using the **copy running-config startup-config** EXEC mode command to ensure consistent ifIndex values.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server ifindex persist**
4. **no snmp-server ifindex persist**
5. **end**

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	**snmp-server ifindex persist** **Example:** `Device(config)# snmp-server ifindex persist`	Globally enables ifIndex values that will remain constant across reboots.
Step 4	**no snmp-server ifindex persist** **Example:** `Device(config)# no snmp-server ifindex persist`	Disables global ifIndex persistence.
Step 5	**end** **Example:** `Device(config)# end`	Exits global configuration mode.

Enabling and Disabling IfIndex Persistence on Specific Interfaces

Perform this task to configure ifIndex persistence only on a specific interface.

Tip Use the **snmp ifindex clear** command on a specific interface when you want that interface to use the global configuration setting for ifIndex persistence. This command clears any ifIndex configuration commands previously entered for that specific interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot* / *port*
4. **snmp ifindex persist**
5. **no snmp ifindex persist**
6. **end**
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type slot / port* **Example:** `Device(config)# interface FastEthernet 0/1`	Enters interface configuration mode for the specified interface. **Note** Note that the syntax of the interface command will vary depending on the platform you are using.
Step 4	**snmp ifindex persist** **Example:** `Device(config-if)# snmp ifindex persist`	Enables an ifIndex value that is constant across reboots on the specified interface.
Step 5	**no snmp ifindex persist** **Example:** `Device(config-if)# no snmp ifindex persist`	Disables an ifIndex value that is constant across reboots on the specified interface.
Step 6	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode.
Step 7	**end** **Example:** `Device(config)# end`	Exits global configuration mode.

Configuring SNMP Support for VPNs

This section describes how to configure SNMP support for VPNs. The SNMP Support for VPNs feature provides configuration commands that allow users to associate SNMP agents and managers with specific VRFs. The specified VRF is used to send SNMP traps and informs and responses between agents and managers. If a VRF is not specified, the default routing table for the VPN is used.

Support for VPNs allows users to configure an SNMP agent to only accept SNMP requests from a certain set of VPNs. With this configuration, providers can provide network management services to their customers who then can manage all user-VPN devices.

Note

- This feature is not supported on all Cisco platforms. Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support.
- Not all MIBs are VPN-aware. To list the VPN-aware MIBs, use the **show snmp mib context** command. For more information about VPN-aware MIBs, see the *SNMP Support over VPNs—Context-based Access Control* configuration module.

Perform this task to configure SNMP support for a specific VPN.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server host** *host-address* [**vrf** *vrf-name*] [**traps** | **informs**] [**version** {**1**| **2c**| **3** [**auth** | **noauth** |**priv**]}] *community-string* [**udp-port** *port*] [*notification-type*]
4. **snmp-server engineID remote** *ip-address* [**udp-port** *udp-port-number*] [**vrf** *vrf-name*] *engineid-string*
5. **exit**
6. **show snmp host**

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	**snmp-server host** *host-address* [**vrf** *vrf-name*] [**traps**	**informs**] [**version** {**1**| **2c**| **3** [**auth**	**noauth** |**priv**]}] *community-string* [**udp-port** *port*] [*notification-type*] **Example:** `Device(config)# snmp-server host example.com public vrf trap-vrf`	Specifies the recipient of an SNMP notification operation and specifies the VRF table to be used for sending SNMP notifications.

Step 4	**snmp-server engineID remote** *ip-address* [**udp-port** *udp-port-number*] [**vrf** *vrf-name*] *engineid-string*
	Example:
	`Device(config)# snmp-server engineID remote 172.16.20.3 vrf traps-vrf`
	Example:
	`80000009030000B064EFE100`
		Configures a name for the remote SNMP engine on a device when configuring SNMP over a specific VPN for a remote SNMP user.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode.
Step 6	**show snmp host** **Example:** `Device# show snmp host`	(Optional) Displays the SNMP configuration and verifies that the SNMP Support for VPNs feature is configured properly.

Configuring Event MIB Using SNMP

The Event MIB can be configured using SNMP directly. In this procedure, the Event MIB is configured to monitor the delta values of ifInOctets for all interfaces once per minute. If any of the samples exceed the specified threshold, a trap notification will be sent.

There are no Cisco software configuration tasks associated with the Event MIB. All configuration of Event MIB functionality must be performed though applications using SNMP. This section provides a sample configuration session using a network management application on an external device. See the "Additional References" section for information about configuring SNMP on your Cisco routing device.

All configuration of Event MIB functionality must be performed though applications using SNMP. The following section provides a step-by-step Event MIB configuration using SNMP research tools available for Sun workstations. The **setany** commands given below are executed using the SNMP application.

Note These are not Cisco command line interface commands. It is assumed that SNMP has been configured on your routing device.

In this configuration, the objective is to monitor ifInOctets for all interfaces. The Event MIB is configured to monitor the delta values of ifInOctets for all interfaces once per minute. If any of the samples exceed the specified threshold of 30, a Trap notification will be sent.

There are five parts to the following example:

Setting the Trigger in the Trigger Table

Perform this task to set the trigger in the trigger table.

SUMMARY STEPS

1. **setany -v2c $ADDRESS private mteTriggerEntryStatus.4.106.111.104.110.1 -i 5**
2. **setany -v2c $ADDRESS private mteTriggerValueID.4.106.111.104.110.1 -d 1.3.6.1.2.1.2.2.1.10**
3. **setany -v2c $ADDRESS private mteTriggerValueIDWildcard.4.106.111.104.110.1 -i 1**
4. **setany -v2c $ADDRESS private mteTriggerTest.4.106.111.104.110.1 -o '20'**
5. **setany -v2c $ADDRESS private mteTriggerFrequency.4.106.111.104.110.1 -g 60**
6. **setany -v2c $ADDRESS private mteTriggerSampleType.4.106.111.104.110.1 -i 2**
7. **setany -v2c $ADDRESS private mteTriggerEnabled.4.106.111.104.110.1 -i 1**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setany -v2c $ADDRESS private mteTriggerEntryStatus.4.106.111.104.110.1 -i 5**	Creates a trigger row in the table with john as the mteOwner and 1 as the trigger name. • The index is given in decimal representation of the ASCII value of john.1.
Step 2	**setany -v2c $ADDRESS private mteTriggerValueID.4.106.111.104.110.1 -d 1.3.6.1.2.1.2.2.1.10**	Sets the mteTriggerValueID to the OID to be watched. • In this example, the OID to be monitored is ifInOctets.
Step 3	**setany -v2c $ADDRESS private mteTriggerValueIDWildcard.4.106.111.104.110.1 -i 1**	Sets the mteTriggerValueIDWildcard to TRUE to denote a object referenced through wildcarding.
Step 4	**setany -v2c $ADDRESS private mteTriggerTest.4.106.111.104.110.1 -o '20'**	Sets the mteTriggerTest to Threshold.
Step 5	**setany -v2c $ADDRESS private mteTriggerFrequency.4.106.111.104.110.1 -g 60**	Sets the mteTriggerFrequency to 60. This means that ifInOctets are monitored once every 60 seconds.
Step 6	**setany -v2c $ADDRESS private mteTriggerSampleType.4.106.111.104.110.1 -i 2**	Sets the sample type to Delta.
Step 7	**setany -v2c $ADDRESS private mteTriggerEnabled.4.106.111.104.110.1 -i 1**	Enables the trigger.

Creating an Event in the Event Table

Perform this task to create an event in the event table.

SUMMARY STEPS

1. **setany -v2c $ADDRESS private mteEventEntryStatus.4.106.111.104.110.101.118.101.110. 116 -i 5**
2. **setany -v2c $ADDRESS private mteEventEnabled.4.106.111.104.110.101.118.101.110.116 -i 1**
3. **setany -v2c $ADDRESS private mteEventEntryStatus.4.106.111.104.110.101.118.101.110. 116 -i 1**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setany -v2c $ADDRESS private mteEventEntryStatus.4.106.111.104.110.101.118.101.110. 116 -i 5**	Creates a row in the Event Table. • The mteOwner here is again john, and the event is mteEventName. • The default action is to send out a notification.
Step 2	**setany -v2c $ADDRESS private mteEventEnabled.4.106.111.104.110.101.118.101.110.116 -i 1**	Enables the Event.

	Command or Action	Purpose
Step 3	**setany -v2c $ADDRESS private mteEventEntryStatus.4.106.111.104.110.101.118.101.110.116 -i 1**	Makes the EventRow active.

Setting and Activating the Trigger Threshold in the Trigger Table

Perform this task to set the trigger threshold in the trigger table.

SUMMARY STEPS

1. **setany -v2c $ADDRESS private mteTriggerThresholdRising.4.106.111.104.110.1 -i 30**
2. **setany -v2c $ADDRESS private mteTriggerThresholdRisingEventOwner.4.106.111.104.110.1 -D "owner"**
3. **setany -v2c $ADDRESS private mteTriggerEntryStatus.4.106.111.104.110.1 -i 1**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setany -v2c $ADDRESS private mteTriggerThresholdRising.4.106.111.104.110.1 -i 30**	Sets the Rising Threshold value to 30. Note that a row would already exist for john.1 in the Trigger Threshold Table.
Step 2	**setany -v2c $ADDRESS private mteTriggerThresholdRisingEventOwner.4.106.111.104.110.1 -D "owner"** **Example:** `setany -v2c $ADDRESS private mteTriggerThresholdRisingEvent.4.106.111.104.110.1 -D "event"`	Points to the entry in the Event Table that specifies the action to be performed.
Step 3	**setany -v2c $ADDRESS private mteTriggerEntryStatus.4.106.111.104.110.1 -i 1**	Makes the trigger active.

What to do next

To confirm that the above configuration is working, ensure that at least one of the interfaces gets more than 30 packets in a minute. This should cause a trap to be sent out after one minute.

Activating the Trigger

Perform this task to activate the trigger.

SUMMARY STEPS

1. **setany -v2c $ADDRESS private mteTriggerEntryStatus.4.106.111.104.110.1 -i 1**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setany -v2c $ADDRESS private mteTriggerEntryStatus.4.106.111.104.110.1 -i 1**	Makes the trigger active.

What to do next

To confirm that the above configuration is working, ensure that at least one of the interfaces gets more than 30 packets in a minute. This should cause a trap to be sent out after one minute.

Monitoring and Maintaining Event MIB

Use the following commands to monitor Event MIB activity from the Cisco command line interface:

Command	Purpose
debug management event mib	Prints messages to the screen whenever the Event MIB evaluates a specified trigger. These messages are given in realtime and are intended to be used by technical support engineers for troubleshooting purposes.
show management event	Displays the SNMP Event values that have been configured on your routing device through the use of the Event MIB.

Configuring Event MIB Using Command Line Interface

The Event MIB can be configured using SNMP directly. In this procedure, the Event MIB is configured to monitor delta values of ifInOctets for all interfaces once per minute. If any of the samples exceed the specified threshold, a trap notification will be sent.

Depending on your release, note that the Event MIB feature is enhanced to add command line interface commands to configure the events, event action, and trigger.

This section contains the following tasks to configure the Event MIB:

Configuring Scalar Variables

Perform this task to configure scalar variables for the Event MIB.

Before you begin

To configure scalar variables for the Event MIB, you should be familiar with the Event MIB scalar variables.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib event sample minimum** *value*
4. **snmp mib event sample instance maximum** *value*
5. **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	**snmp mib event sample minimum** *value* **Example:** `Device(config)# snmp mib event sample minimum 10`	Sets the minimum value for object sampling.
Step 4	**snmp mib event sample instance maximum** *value* **Example:** `Device(config)# snmp mib event sample instance maximum 50`	Sets the maximum value for object instance sampling.
Step 5	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode.

Configuring Event MIB Object List

To configure the Event MIB, you need to set up a list of objects that can be added to notifications according to the trigger, trigger test, or event.

Before you begin

To configure the Event MIB object list, you should be familiar with the Event MIB objects and object identifiers, which can be added to notifications according to the event, trigger, or trigger test.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib event object list owner** *object-list-owner* **name** *object-list-name object-number*
4. **object id** *object-identifier*
5. **wildcard**
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**snmp mib event object list owner** *object-list-owner* **name** *object-list-name object-number* **Example:** `Device(config)# snmp mib event object list owner owner1 name objectA 10`	Configures the Event MIB object list.
Step 4	**object id** *object-identifier* **Example:** `Device(config-event-objlist)# object id ifInOctets`	Specifies the object identifier for the object configured for the event.
Step 5	**wildcard** **Example:** `Device(config-event-objlist)# wildcard`	(Optional) Starts a wildcard search for object identifiers. By specifying a partial object identifier, you can obtain a list of object identifiers.
Step 6	**end** **Example:** `Device(config-event-objlist)# end`	Exits object list configuration mode.

Configuring Event

Perform this task to configure a management event.

Before you begin

To configure a management event, you should be familiar with the SNMP MIB events and object identifiers.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib event owner** *event-owner* **name** *event-name*
4. **description** *event-description*
5. **enable**
6. **end**

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	**snmp mib event owner** *event-owner* **name** *event-name* **Example:** `Device(config)# snmp mib event owner owner1 name EventA`	Enters the event configuration mode.
Step 4	**description** *event-description* **Example:** `Device(config-event)# description "EventA is an RMON event"`	Describes the function and use of the event.
Step 5	**enable** **Example:** `Device(config-event)# enable`	Enables the event. **Note** The event can be executed during an event trigger only if it is enabled.
Step 6	**end** **Example:** `Device(config-event)# end`	Exits event configuration mode and returns to privileged EXEC mode.

Configuring Event Action

By configuring an event action, you can define the actions that an application can perform during an event trigger. The actions for an event include sending a notification, setting a MIB object and so on. You can set the event action information to either **set** or **notification**. The actions for the event can be configured only in event configuration mode.

The following sections contain the tasks to configure an event action:

Configuring Action Notification

Perform this task to set the notification action for the event.

SUMMARY STEPS

1. **enable**

2. **configure terminal**
3. **snmp mib event owner** *event-owner* **name** *event-name*
4. **action notification**
5. **object id** *object-id*
6. **end**

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	**snmp mib event owner** *event-owner* **name** *event-name* **Example:** `Device(config)# snmp mib event owner owner1 event EventA`	Enters event configuration mode.
Step 4	**action notification** **Example:** `Device(config-event)# action notification`	Sets the notification action for an event. **Note** If the event action is set to notification, a notification is generated whenever an object associated with an event is modified.
Step 5	**object id** *object-id* **Example:** `Device(config-event-action-notification)# object id ifInOctets`	Configures object for action notification. When the object specified is modified, a notification will be sent to the host system.
Step 6	**end** **Example:** `Device(config-event-action-notification)# end`	Exits action notification configuration mode and returns to privileged EXEC mode.

Configuring Action Set

Perform this task to set actions for an event.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **action set**
4. **object id** *object-id*

5. **value** *integer-value*
6. **end**

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	**action set** **Example:** `Device(config-event)# action set`	Enters action set configuration mode.
Step 4	**object id** *object-id* **Example:** `Device(config-event-action-set)# object id ifInOctets`	Configures object for action set. When the object specified is modified, a specified action will be performed.
Step 5	**value** *integer-value* **Example:** `Device(config-event-action-set)# value 10`	Sets a value for the object.
Step 6	**end** **Example:** `Device(config-event-action-set)# end`	Exits action set configuration mode and returns to privileged EXEC mode.

Configuring Event Trigger

By configuring an event trigger, you can list the objects to monitor, and associate each trigger to an event.

Perform this task to configure an event trigger.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib event trigger owner** *trigger-owner* **name** *trigger-name*
4. **description** *trigger-description*
5. **frequency** *seconds*

6. **object list owner** *object-list-owner* **name** *object-list-name*
7. **object id** *object-identifier*
8. **enable**
9. **end**

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	**snmp mib event trigger owner** *trigger-owner* **name** *trigger-name* **Example:** `Device(config)# snmp mib event trigger owner owner1 name EventTriggerA`	Enables event trigger configuration mode for the specified event trigger.
Step 4	**description** *trigger-description* **Example:** `Device(config-event-trigger)# description "EventTriggerA is an RMON alarm."`	Describes the function and use of the event trigger.
Step 5	**frequency** *seconds* **Example:** `Device(config-event-trigger)# frequency 120`	Configures the waiting time (number of seconds) between trigger samples.
Step 6	**object list owner** *object-list-owner* **name** *object-list-name* **Example:** `Device(config-event-trigger)# object list owner owner1 name ObjectListA`	Specifies the list of objects that can be added to notifications.
Step 7	**object id** *object-identifier* **Example:** `Device(config-event-trigger)# object id ifInOctets`	Configures object identifiers for an event trigger.

	Command or Action	Purpose
Step 8	**enable** **Example:** `Device(config-event-trigger)# enable`	Enables the event trigger.
Step 9	**end** **Example:** `Device(config-event-trigger)# end`	Exits event trigger configuration mode.

Configuring Existence Trigger Test

You should configure this trigger type in event trigger configuration mode.

Perform this task to configure trigger parameters for the test existence trigger type.

SUMMARY STEPS

1. **test existence**
2. **event owner** *event-owner* **name** *event-name*
3. **object list owner** *object-list-owner* **name** *object-list-name*
4. **type** {**present** | **absent** | **changed**}
5. **startup** {**present** | **absent**}
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**test existence** **Example:** `Device(config-event-trigger)# test existence`	Enables test existence configuration mode.
Step 2	**event owner** *event-owner* **name** *event-name* **Example:** `Device(config-event-trigger-existence)# event owner owner1 name EventA`	Configures the event for the existence trigger test.
Step 3	**object list owner** *object-list-owner* **name** *object-list-name* **Example:** `Device(config-event-trigger-existence)# object list owner owner1 name ObjectListA`	Configures the list of objects for the existence trigger test.
Step 4	**type** {**present**	**absent**	**changed**} **Example:**	Performs the specified type of existence test. Existence tests are of the following three types:

	Command or Action	Purpose
	`Device(config-event-trigger-existence)# type present`	• Present—Setting type to present tests if the objects that appear during the event trigger exist. • Absent—Setting type to absent tests if the objects that disappear during the event trigger exist. • Changed—Setting type to changed tests if the objects that changed during the event trigger exist.
Step 5	**startup** {**present** | **absent**} **Example:** `Device(config-event-trigger-existence)# startup present`	Triggers an event if the test is performed successfully.
Step 6	**end** **Example:** `Device(config-event-trigger-existence)# end`	Exits existence trigger test configuration mode.

Configuring Boolean Trigger Test

You should configure this trigger test in event trigger configuration mode.

Perform this task to configure trigger parameters for the Boolean trigger type.

SUMMARY STEPS

1. **test boolean**
2. **comparison** {**unequal** | **equal** | **less** | **lessOrEqual** | **greater** | **greaterOrEqual**}
3. **value** *integer-value*
4. **object list owner** *object-list-owner* **name** *object-list-name*
5. **event owner** *event-owner* **name** *event-name*
6. **startup**
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**test boolean** **Example:** `Device(config-event-trigger)# test boolean`	Enables Boolean trigger test configuration mode.
Step 2	**comparison** {**unequal** | **equal** | **less** | **lessOrEqual** | **greater** | **greaterOrEqual**} **Example:** `Device(config-event-trigger-boolean)# comparison unequal`	Performs the specified Boolean comparison test. • The value for the Boolean comparison test can be set to unequal, equal, less, lessOrEqual, greater, or greaterOrEqual.

	Command or Action	Purpose
Step 3	**value** *integer-value* **Example:** `Device(config-event-trigger-boolean)# value 10`	Sets a value for the Boolean trigger test.
Step 4	**object list owner** *object-list-owner* **name** *object-list-name* **Example:** `Device(config-event-trigger-boolean)# object list owner owner1 name ObjectListA`	Configures the list of objects for the Boolean trigger test.
Step 5	**event owner** *event-owner* **name** *event-name* **Example:** `Device(config-event-trigger-boolean)# event owner owner1 name EventA`	Configures the event for the Boolean trigger type.
Step 6	**startup** **Example:** `Device(config-event-trigger-boolean)# startup`	Triggers an event if the test is performed successfully.
Step 7	**end** **Example:** `Device(config-event-trigger-boolean)# end`	Exits Boolean trigger test configuration mode.

Configuring Threshold Trigger Test

You should configure this trigger test in event trigger configuration mode.

Perform this task to configure trigger parameters for the threshold trigger test.

SUMMARY STEPS

1. **test threshold**
2. **object list owner** *object-list-owner* **name** *object-list-name*
3. **rising** *integer-value*
4. **rising event owner** *event-owner* **name** *event-name*
5. **falling** *integer-value*
6. **falling event owner** *event-owner* **name** *event-name*
7. **delta rising** *integer-value*
8. **delta rising event owner** *event-owner* **name** *event-name*
9. **delta falling** *integer-value*
10. **delta falling event owner** *event-owner* **name** *event-name*
11. **startup** {**rising** | **falling** | **rising-or-falling**}
12. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**test threshold** **Example:** `Device(config-event-trigger)# test threshold`	Enables threshold trigger test configuration mode.
Step 2	**object list owner** *object-list-owner* **name** *object-list-name* **Example:** `Device(config-event-trigger-threshold)# object list owner owner1 name ObjectListA`	Configures the list of objects for the threshold trigger test.
Step 3	**rising** *integer-value* **Example:** `Device(config-event-trigger-threshold)# rising 100`	Sets the rising threshold to the specified value.
Step 4	**rising event owner** *event-owner* **name** *event-name* **Example:** `Device(config-event-trigger-threshold)# rising event owner owner1 name EventA`	Configures an event for the threshold trigger test for the rising threshold.
Step 5	**falling** *integer-value* **Example:** `Device(config-event-trigger-threshold)# falling 50`	Sets the falling threshold to the specified value.
Step 6	**falling event owner** *event-owner* **name** *event-name* **Example:** `Device(config-event-trigger-threshold)# falling event owner owner1 name EventB`	Configures an event for the threshold trigger test for the falling threshold.
Step 7	**delta rising** *integer-value* **Example:** `Device(config-event-trigger-threshold)# delta rising 30`	Sets the delta rising threshold to the specified value when the sampling method specified for the event trigger is delta.
Step 8	**delta rising event owner** *event-owner* **name** *event-name* **Example:** `Device(config-event-trigger-threshold)# delta rising event owner owner1 name EventC`	Configures an event for the threshold trigger test for the delta rising threshold.
Step 9	**delta falling** *integer-value* **Example:** `Device(config-event-trigger-threshold)# delta falling 10`	Sets the delta falling threshold to the specified value when the sampling method specified for the event trigger is delta.

	Command or Action	Purpose
Step 10	**delta falling event owner** *event-owner* **name** *event-name* **Example:** `Device(config-event-trigger-threshold)# delta falling event owner owner1 name EventAA`	Configures an event for the threshold target test for the delta falling threshold.
Step 11	**startup** {**rising** | **falling** | **rising-or-falling**} **Example:** `Device(config-event-trigger-threshold)# startup rising`	Triggers an event when the threshold trigger test conditions are met.
Step 12	**end** **Example:** `Device(config-event-trigger-threshold)# end`	Exits threshold trigger test configuration mode.

Configuring Expression MIB Using SNMP

Expression MIB can be configured using SNMP directly.

There are no Cisco software configuration tasks associated with Expression MIB. All configurations of the Expression MIB functionality must be performed though applications using SNMP. This section provides a sample configuration session using a network management application on an external device. See the Additional References section for information about configuring SNMP on your Cisco routing device.

The following section provides a step-by-step Expression MIB configuration using SNMP research tools available for Sun workstations. The **setany** commands given below are executed using the SNMP application. Note that these commands are not Cisco command line interface commands. It is assumed that SNMP has been configured on your routing device.

In the following configuration, a wildcarded expression involving the addition of the counters ifInOctects and ifOutOctects are evaluated.

SUMMARY STEPS

1. **setany -v2c $SNMP_HOST private expResourceDeltaMinimum.0 -i 60**
2. **setany -v2c $SNMP_HOST private expExpressionIndex.116.101.115.116 -g 9**
3. **setany -v2c $SNMP_HOST private expNameStatus.116.101.115.116 -i 5**
4. **setany -v2c $SNMP_HOST private expExpressionComment.9 -D "test expression"**
5. **setany -v2c $SNMP_HOST private expExpression.9 -D '$1 + $2'**
6. **setany -v2c $SNMP_HOST private expObjectID.9.1 -d ifInOctets**
7. **setany -v2c $SNMP_HOST private expObjectSampleType.9.1 -i 2**
8. **setany -v2c $SNMP_HOST private expObjectIDWildcard.9.1 -i 1**
9. **setany -v2c $SNMP_HOST private expObjectStatus.9.1 -i 1**
10. **setany -v2c $SNMP_HOST private expNameStatus.116.101.115.116 -i 1**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setany -v2c $SNMP_HOST private expResourceDeltaMinimum.0 -i 60**	Sets the minimum delta interval that the system will accept.
Step 2	**setany -v2c $SNMP_HOST private expExpressionIndex.116.101.115.116 -g 9**	Sets the identification number used for identifying the expression. • For example, expName can be 'test', which is ASCII 116.101.115.116.
Step 3	**setany -v2c $SNMP_HOST private expNameStatus.116.101.115.116 -i 5**	Creates an entry in the expNameStatusTable. **Note** When an entry is created in the expNameTable, it automatically creates an entry in the expExpressionTable.
Step 4	**setany -v2c $SNMP_HOST private expExpressionComment.9 -D "test expression"**	Sets the object to a comment to explain the use or meaning of the expression. • Here, the comment is "test expression".
Step 5	**setany -v2c $SNMP_HOST private expExpression.9 -D '$1 + $2'**	Sets the object expExpression to an expression that needs to be evaluated. • In this expression, "$1" corresponds to "ifInOctets", "$2" corresponds to "ifOutOctets", and the expression signifies the addition of the two counter objects.
Step 6	**setany -v2c $SNMP_HOST private expObjectID.9.1 -d ifInOctets** **Example:** `setany -v2c $SNMP_HOST private expObjectID.9.2 -d ifOutOctets`	Specifies the object identifiers used in the expression mentioned in the above set for calculation. • Here, the number "9", suffixed to the object expObjectID, corresponds to the unique identifier used for identifying the expression, and the number "1" following "9" is another unique identifier used for identifying an object within the expression. Set the expObjectID to the two objects used in forming the expression.
Step 7	**setany -v2c $SNMP_HOST private expObjectSampleType.9.1 -i 2** **Example:** `setany -v2c $SNMP_HOST private expObjectSampleType.9.2 -i 2`	Sets the type of sampling to be done for objects in the expression. • There are two types of sampling: a) Absolute b) Delta. Here, the sample type has been set to "Delta".
Step 8	**setany -v2c $SNMP_HOST private expObjectIDWildcard.9.1 -i 1** **Example:** `setany -v2c $SNMP_HOST private expObjectIDWildcard.9.2 -i 1`	Specifies whether the expObjectID is wildcarded or not. In this case, both the expObjectID are wildcarded.

	Command or Action	Purpose
Step 9	**setany -v2c $SNMP_HOST private expObjectStatus.9.1 -i 1** **Example:** `setany -v2c $SNMP_HOST private expObjectStatus.9.2 -i 1`	Sets the rows in the expObjectTable to active.
Step 10	**setany -v2c $SNMP_HOST private expNameStatus.116.101.115.116 -i 1**	Sets the rows in the expNameTable to active so that the value of the expression can be evaluated. • The value of the expression can now be obtained from the expValueTable.

Configuring Expression MIB Using the CLI

Expression MIB can be configured using SNMP directly. However, in Cisco IOS Release 12.4(20)T, the Expression MIB feature is enhanced to add CLIs to configure expressions. You should be familiar with expressions, object identifiers, and sampling methods before configuring Expression MIB.

The following sections contain the tasks to configure Expression MIB:

Configuring Expression MIB Scalar Objects

Expression MIB has the following scalar objects:

- expResourceDeltaMinimum
- expResourceDeltaWildcardInstanceMaximum

Perform this task to configure Expression MIB scalar objects.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib expression delta minimum** *seconds*
4. **snmp mib expression delta wildcard maximum** *number-of-instances*
5. **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:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**snmp mib expression delta minimum** *seconds* **Example:** `Device(config)# snmp mib expression delta minimum 20`	(Optional) Sets the minimum delta interval in seconds. **Note** Application may use larger values for this minimum delta interval to lower the impact of constantly computing deltas. For larger delta sampling intervals, the application samples less often and has less overhead. By using this command, you can enforce a lower overhead for all expressions created after the delta interval is set.
Step 4	**snmp mib expression delta wildcard maximum** *number-of-instances* **Example:** `Device(config)# snmp mib expression delta wildcard maximum 120`	(Optional) Limits the maximum number of dynamic instance entries for wildcarded delta objects in expressions. For a given delta expression, the number of dynamic instances is the number of values that meet all criteria to exist, times the number of delta values in the expression. There is no preset limit for the instance entries and it is dynamic based on a system's resources.
Step 5	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring Expressions

Perform this task to configure an expression.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib expression owner** *expression-owner* **name** *expression-name*
4. **description** *expression-description*
5. **expression** *expression*
6. **delta interval** *seconds*
7. **value type** {**counter32** | **unsigned32** | **timeticks** | **integer32** | **ipaddress** | **octetstring** | **objectid** | **counter64**}
8. **enable**
9. **object** *object-number*
10. **id** *object-identifier*
11. **wildcard**
12. **discontinuity object** *discontinuity-object-id* [**wildcard**] [**type** {**timeticks** | **timestamp** | **date-and-time**}]
13. **conditional object** *conditional-object-id* **[wildcard]**
14. **sample** {**absolute** | **delta** | **changed**}

15. **end**

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	**snmp mib expression owner** *expression-owner* **name** *expression-name* **Example:** `Device(config-expression)# snmp mib expression owner owner1 name ExpA`	Enables the expression to be configured.
Step 4	**description** *expression-description* **Example:** `Device(config-expression)# description this expression is created for the sysLocation MIB object`	Configures a description for the expression.
Step 5	**expression** *expression* **Example:** `Device(config-expression)# expression ($1+$2)*800/$3`	Configures the expression to be evaluated. **Note** The expressions are in ANSI C syntax. However, the variables in an expression are defined as a combination of the dollar sign ($) and an integer that corresponds to the object number of the object used in evaluating the expression.
Step 6	**delta interval** *seconds* **Example:** `Device(config-expression)# delta interval 180`	Configures the sampling interval for objects in the expression if the sampling method is delta.
Step 7	**value type** {**counter32** | **unsigned32** | **timeticks** | **integer32** | **ipaddress** | **octetstring** | **objectid** | **counter64**} **Example:** `Device(config-expression)# value type counter32`	Sets the specified value type for the expression.
Step 8	**enable** **Example:** `Device(config-expression)# enable`	Enables an expression for evaluation.

	Command or Action	Purpose
Step 9	**object** *object-number* **Example:** `Device(config-expression)# object 2`	Configures the objects that are used for evaluating an expression. • The object number is used to associate the object with the variables in the expression. The variable corresponding to the object is $ and object number. Thus, the variable in the example used here corresponds to $10.
Step 10	**id** *object-identifier* **Example:** `Device(config-expression-object)# id ifInOctets`	Configures the object identifier.
Step 11	**wildcard** **Example:** `Device(config-expression-object)# wildcard`	(Optional) Enables a wildcarded search for objects used in evaluating an expression.
Step 12	**discontinuity object** *discontinuity-object-id* [**wildcard**] [**type** {**timeticks** | **timestamp** | **date-and-time**}] **Example:** `Device(config-expression-object)# discontinuity object sysUpTime`	(Optional) Configures the discontinuity properties for the object if the object sampling type is set to delta or changed. The discontinuity object ID supports normal checking for a discontinuity in a counter. • Using the **wildcard** keyword, you can enable wildcarded search for objects with discontinuity properties. • Using the **type** keyword, you can set value for objects with discontinuity properties.
Step 13	**conditional object** *conditional-object-id* **[wildcard]** **Example:** `Device(config-expression-object)# conditional object mib-2.90.1.3.1.1.2.3.112.99.110.4.101.120.112.53`	(Optional) Configures the conditional object identifier. • Using the **wildcard** keyword, you can enable a wildcarded search for conditional objects with discontinuity properties.
Step 14	**sample** {**absolute** | **delta** | **changed**} **Example:** `Device(config-expression-object)# sample delta`	Enables the specified sampling method for the object. This example uses the delta sampling method. You can set any of the three sampling methods: absolute, delta, and changed. • Absolute sampling—Uses the value of the MIB object during sampling. • Delta sampling—Uses the last sampling value maintained in the application. This method requires applications to do continuous sampling. • Changed sampling—Uses the changed value of the object since the last sample.

	Command or Action	Purpose
Step 15	**end** **Example:** `Device(config-expression-object)# end`	Exits expression object configuration mode.

Configuration Examples for SNMP Support

Example Configuring SNMPv1, SNMPv2c and SNMPv3

The following example shows how to enable SNMPv1, SNMPv2c, and SNMPv3. The configuration permits any SNMP manager to access all objects with read-only permissions using the community string named public. This configuration does not cause the device to send traps.

```
Device(config)# snmp-server community public
```

The following example shows how to permit SNMP access to all objects with read-only permission using the community string named public. The device will also send ISDN traps to the hosts 172.16.1.111 and 172.16.1.33 using SNMPv1 and to the host 172.16.1.27 using SNMPv2c. The community string named public is sent with the traps.

```
Device(config)# snmp-server community public
Device(config)# snmp-server enable traps isdn
Device(config)# snmp-server host 172.16.1.27 version 2c public
Device(config)# snmp-server host 172.16.1.111 version 1 public
Device(config)# snmp-server host 172.16.1.33 public
```

The following example shows how to allow read-only access for all objects to members of access list 4 that specify the comaccess community string. No other SNMP managers have access to any objects. SNMP Authentication Failure traps are sent by SNMPv2c to the host example.com using the community string named public.

```
Device(config)# snmp-server community comaccess ro 4
Device(config)# snmp-server enable traps snmp authentication
Device(config)# snmp-server host example.com version 2c public
```

The following example shows how to configure a remote user to receive traps at the noAuthNoPriv security level when the SNMPv3 security model is enabled:

```
Device(config)# snmp-server group group1 v3 noauth
Device(config)# snmp-server user remoteuser1 group1 remote 10.12.8.4
Device(config)# snmp-server host 10.12.8.4 informs version 3 noauth remoteuser config
```

The following example shows how to configure a remote user to receive traps at the authNoPriv security level when the SNMPv3 security model is enabled:

```
Device(config)# snmp-server group group2 v3 auth
Device(config)# snmp-server user AuthUser group2 remote 10.12.8.4 v3 auth md5 password1
```

The following example shows how to configure a remote user to receive traps at the priv security level when the SNMPv3 security model is enabled:

```
Device(config)# snmp-server group group3 v3 priv
Device(config)# snmp-server user PrivateUser group3 remote 10.12.8.4 v3 auth md5 password1
 priv access des56
```

The following example shows how to send Entity MIB inform notifications to the host example.com. The community string is restricted. The first line enables the device to send Entity MIB notifications in addition to any traps or informs previously enabled. The second line specifies that the notifications should be sent as informs, specifies the destination of these informs, and overwrites the previous **snmp-server host** commands for the host example.com.

```
Device(config)# snmp-server enable traps entity
Device(config)# snmp-server host informs example.com restricted entity
```

The following example shows how to send SNMP and Cisco environmental monitor enterprise-specific traps to the address 172.30.2.160:

```
Device(config)# snmp-server enable traps
Device(config)# snmp-server host 172.30.2.160 public snmp envmon
```

The following example shows how to enable the device to send all traps to the host example.com using the community string public:

```
Device(config)# snmp-server enable traps
Device(config)# snmp-server host example.com public
```

The following example shows a configuration in which no traps are sent to a host. The BGP traps are enabled for all hosts, but only the ISDN traps are enabled to be sent to a host.

```
Device(config)# snmp-server enable traps bgp
Device(config)# snmp-server host host1 public isdn
```

The following example shows how to enable a device to send all informs to the host example.com using the community string named public:

```
Device(config)# snmp-server enable traps
Device(config)# snmp-server host example.com informs version 2c public
```

In the following example, the SNMP manager is enabled and the session timeout is set to a value greater than the default:

```
Device(config)# snmp-server manager
Device(config)# snmp-server manager session-timeout 1000
```

Example Configuring IfAlias Long Name Support

In the following example a long description is applied to the Fast Ethernet interface in slot 1, port adapter 0, and port 0:

```
Device# configure terminal
Device(config)#interface FastEthernet1/0/0
Device(config-if)# description FastEthernet1/0/0 this is a test of a description that exceeds
 64 characters in length
Device(config-if)#ip address 192.168.134.55 255.255.255.0
Device(config-if)#no ip directed-broadcast
```

```
Device(config-if)#no ip route-cache distributed
```

Assuming that ifAlias long name support is not yet enabled (the default), the following example shows the results of a mibwalk operation from an NMS:

```
***** SNMP QUERY STARTED *****
 .
 .
 .
 ifXEntry.18.10 (octets) (zero-length)
 ifXEntry.18.11 (octets) Fastethernet1/0/0 this is a test of a description that exceeds 64
 ch
 ifXEntry.18.12 (octets) (zero-length)
 .
 .
 .
```

The following output shows the description that is displayed at the CLI:

```
Device# show interface FastEthernet0/0/0

FastEthernet1/0/0 is administratively down, line protocol is down
  Hardware is Lance, address is 0010.7b4d.7046 (bia 0010.7b4d.7046)
  Description: FastEthernet1/0/0 this is a test of a description that exceeds 64 chh
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
     reliability 252/255, txload 1/255, rxload 1/255
 .
 .
 .
```

In the following example, ifAlias long name support is enabled and the description is displayed again:

```
Device(config)# snmp ifmib ifalias long
Device(config)#interface FastEthernet1/0/0
Device(config-if)# description FastEthernet1/0/0 this is a test of a description that exceeds
64 characters in length
Device(config)#end

Device# show interface FastEthernet1/0/0

FastEthernet1/0/0 is administratively down, line protocol is down
  Hardware is Lance, address is 0010.7b4d.7046 (bia 0010.7b4d.7046)
  Description: FastEthernet1/0/0 this is a test of a description that exceeds 64 characters
 in length
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
     reliability 252/255, txload 1/255, rxload 1/255
 .
 .
 .
***** SNMP QUERY STARTED *****
 .
 .
 .
 ifXEntry.18.10 (octets) (zero-length)
 ifXEntry.18.11 (octets) FastEthernet1/0/0 this is a test of a description that exceeds 64
 characters in length
 ifXEntry.18.12 (octets) (zero-length)
 .
 .
 .
```

Example Configuring SNMP Support for VPNs

In the following example, all SNMP notifications are sent to example.com over the VRF named trap-vrf:

```
Device(config)# snmp-server host example.com vrf trap-vrf
```

In the following example, the VRF named "traps-vrf" is configured for the remote server 172.16.20.3:

```
Device(config)# snmp-server engineID remote 172.16.20.3 vrf traps-vrf 80000009030000B064EFE100
```

Example Configuring Event MIB

The following example shows how to configure scalar variables for an event:

```
Device# configure terminal
Device(config)# snmp mib event sample minimum 10
Device(config)# snmp mib event sample instance maximum 50
Device(config)# exit
```

The following example shows how to configure the object list for an event:

```
Device# configure terminal
Device(config)# snmp mib event object list owner owner1 name objectA 1
Device(config-event-objlist)# object id ifInOctets
Device(config-event-objlist)# wildcard
Device(config-event-objlist)# exit
```

The following example shows how to configure an event:

```
Device# configure terminal
Device(config)# snmp mib event owner owner1 name EventA
Device(config-event)# description "eventA is an RMON event."
Device(config-event)# enable
Device(config-event)# exit
```

The following example shows how to set the notification action for an event:

```
Device(config-event)# action notification
Device(config-event-action-notification)# object id ifInOctets
Device(config-event-action-notification)# exit
```

The following example shows how to set actions for an event:

```
Device(config-event)# action set
Device(config-event-action-set)# object id ifInOctets
Device(config-event-action-set)# value 10
Device(config-event-action-set)# exit
```

The following example shows how to configure the trigger for an event:

```
Device# configure terminal
Device(config)# snmp mib event trigger owner owner1 name EventTriggerA
Device(config-event-trigger)# description “EventTriggerA is an RMON alarm.”
Device(config-event-trigger)# frequency 120
Device(config-event-trigger)# object list owner owner1 name ObjectListA
Device(config-event-trigger)# object id ifInOctets
Device(config-event-trigger-object-id)# enable
Device(config-event-trigger)# exit
```

The following example shows how to configure the existence trigger test:

```
Device(config-event-trigger)# test existence
Device(config-event-trigger-existence)# event owner owner1 name EventA
Device(config-event-trigger-existence)# object list owner owner1 name ObjectListA
Device(config-event-trigger-existence)# type present
```

```
Device(config-event-trigger-existence)# startup present
Device(config-event-trigger-existence)# exit
```

The following example shows how to configure the Boolean trigger test:

```
Device(config-event-trigger)# test boolean
Device(config-event-trigger-boolean)# comparison unequal
Device(config-event-trigger-boolean)# value 10
Device(config-event-trigger-boolean)# object list owner owner1 name ObjectListA
Device(config-event-trigger-boolean)# event owner owner1 name EventA
Device(config-event-trigger-boolean)# startup
Device(config-event-trigger-boolean)# exit
```

The following example shows how to configure the threshold trigger test:

```
Device(config-event-trigger)# test threshold
Device(config-event-trigger-threshold)# object list owner owner1 name ObjectListA
Device(config-event-trigger-threshold)# rising 100
Device(config-event-trigger-threshold)# rising event owner owner1 name EventA
Device(config-event-trigger-threshold)# falling 50
Device(config-event-trigger-threshold)# falling event owner owner1 name EventA
Device(config-event-trigger-threshold)# delta rising 30
Device(config-event-trigger-threshold)# delta rising event owner owner1 name EventA
Device(config-event-trigger-threshold)# delta falling 10
Device(config-event-trigger-threshold)# delta falling event owner owner1 name EventA
Device(config-event-trigger-threshold)# startup rising
Device(config-event-trigger-threshold)# exit
```

Example Configuring Expression MIB

The following example shows how to configure Expression MIB using the **snmp mib expression**command in global configuration mode:

```
Device(config)# snmp mib expression owner pcn name exp6

Device(config-expression)# description this expression is created for the
sysLocation MIB object

Device(config-expression)# expression ($1+$2)*800/$3

Device(config-expression)# delta interval 120

Device(config-expression)# value type counter32

Device(config-expression)# enable

Device(config-expression)# object 2

Device(config-expression-object)# id ifInOctets

Device(config-expression-object)# wildcard

Device(config-expression-object)# discontinuity object sysUpTime
```

```
Device(config-expression-object)# conditional object
mib-2.90.1.3.1.1.2.3.112.99.110.4.101.120.112.53 wildcard

Device(config-expression-object)# sample delta

Device(config-expression-object)# end
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Cisco IOS SNMP Support Command Reference	Cisco IOS SNMP Support Command Reference

Standards and RFCs

Standard/RFC	Title
CBC-DES (DES-56) standard	*Symmetric Encryption Protocol*
Standard 58	*Structure of Management Information Version 2 (SMIv2) >*
RFC 1067	*A Simple Network Management Protocol*
RFC 1091	*Telnet terminal-type option*
RFC 1098	*Simple Network Management Protocol (SNMP)*
RFC 1157	*Simple Network Management Protocol (SNMP)*
RFC 1213	*Management Information Base for Network Management of TCP/IP-based internets:MIB-II*
RFC 1215	*Convention for defining traps for use with the SNMP*
RFC 1901	*Introduction to Community-based SNMPv2*
RFC 1905	*Common Management Information Services and Protocol over TCP/IP (CMOT)*
RFC 1906	*Telnet X Display Location Option*
RFC 1908	*Simple Network Management Protocol (SNMP)*
RFC 2104	*HMAC: Keyed-Hashing for Message Authentication*
RFC 2206	*RSVP Management Information Base using SMIv2*

Standard/RFC	Title
RFC 2213	*Integrated Services Management Information Base using SMIv2*
RFC 2214	*Integrated Services Management Information Base Guaranteed Service Extensions using SMIv2*
RFC 2233	The Interface Group MIB using SMIv2
RFC 2271	*An Architecture for Describing SNMP Management Frameworks*
RFC 2570	*Introduction to Version 3 of the Internet-standard Network Management Framework*
RFC 2578	*Structure of Management Information Version 2 (SMIv2)*
RFC 2579	*Textual Conventions for SMIv2*
RFC 2580	*Conformance Statements for SMIv2*
RFC 2981	Event MIB
RFC 3413	*SNMPv3 Applications*
RFC 3415	*View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)*

MIBs

MIB	MIBs Link
• Cisco SNMPv2 • Ethernet-like Interfaces MIB • Event MIB • Expression MIB Support for Delta, Wildcarding, and Aggregation • Interfaces Group MIB (IF-MIB) • Interfaces Group MIB Enhancements • MIB Enhancements for Universal Gateways and Access Servers	To locate and download MIBs for selected platforms, Cisco IOS XE software 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/cisco/web/support/index.html

Feature Information for Configuring SNMP Support

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 Configuring SNMP Support

Feature Name	Releases	Feature Information
Event MIB	Cisco IOS XE Release 2.1	The Event MIB feature was implemented on the Cisco ASR 1000 series routers.

Feature Name	Releases	Feature Information
Event MIB and Expression MIB CLIs	Cisco IOS XE Release 3.1S	The Event MIB and Expression MIB feature introduces CLIs to configure the Event MIB and Expression MIB. The following commands were introduced by this feature: **action (event)** , **comparison**, **conditional object**, **delta (test threshold)**, **delta interval**, **description (event)**, **description (expression)**, **description (trigger)**, **discontinuity object**, **enable (event)**, **enable (expression)**, **event owner**, **enable (expression)**, **expression**, **falling (test threshold)**, **frequency (event trigger)**, **object (expression)**, **object-id (action notification)**, **object id (action set)**, **object id (event trigger)**, **object list (trigger test)**, **object wildcard**, **rising (test threshold)**, **sample (expression)**, **snmp mib event object list**, **snmp mib event owner**, **snmp mib event trigger**, **snmp mib expression delta**, **snmp mib expression owner**, **startup (test existence)**, **startup (test boolean)**, **startup (test threshold)**, **test (event trigger)**, **type (test existence)**, **value (test boolean)**, **value (event configuration)**, **value type**, **wildcard (event and expression)**.
Interface Index Display for SNMP	Cisco IOS XE Release 2.1	The Interface Index Display for SNMP feature introduces new commands and command modifications that allow advanced users of SNMP to view information about the interface registrations directly on the managed agent. You can display MIB information from the agent without using an external NMS. This feature addresses three objects in the Interfaces MIB: *ifIndex* , *ifAlias* , and *ifName* . For complete definitions of these objects, see the IF-MIB.my file available from the Cisco SNMPv2 MIB website at ftp://ftp.cisco.com/pub/mibs/v2/.
Interface Index Persistence	Cisco IOS XE Release 2.1	The Interface Index Persistence feature enhancement allows interfaces to be identified with unique values which will remain constant even when a device is rebooted. These interface identification values are used for network monitoring and management using SNMP.
SNMP (Simple Network Management Protocol)	Cisco IOS XE Release 2.1	
SNMP Diagnostics	Cisco IOS XE Release 3.1S	The SNMP Diagnostics feature adds Cisco IOS CLI commands to display the object identifiers that are recently requested by the network management system, and to display the SNMP debug messages. The following commands were introduced or modified: **show snmp stats oid** and **debug snmp detail**.

Feature Name	Releases	Feature Information
SNMP Inform Request	Cisco IOS XE Release 2.1	
SNMP Manager	Cisco IOS XE Release 2.1	The SNMP Manager feature was implemented on the Cisco ASR 1000 series routers.
SNMP Notification Logging	Cisco IOS XE Release 2.1	The SNMP Notification Logging feature adds Cisco IOS CLI commands to change the size of the notification log, to set the global ageout value for the log, and to display logging summaries at the command line.
SNMP Support for VPNs	Cisco IOS XE Release 2.1	The SNMP Support for VPNs feature allows SNMP traps and informs to be sent and received using VRF tables. In particular, this feature adds support to Cisco IOS XE software for sending and receiving SNMP traps and informs specific to individual VPNs.
SNMP Version 3	Cisco IOS XE Release 2.1	
SNMPv2C	Cisco IOS XE Release 2.1	

Glossary

ifAlias—SNMP Interface Alias. The ifAlias is an object in the IF-MIB. The ifAlias is an alias name for the interface as specified by the network manager that provides a nonvolatile description for the interface. For a complete definition, see the IF-MIB.my file.

ifIndex—SNMP Interface Index. The ifIndex is an object in the IF-MIB. The ifIndex is a unique integer assigned to every interface (including subinterfaces) on the managed system when the interface registers with the IF-MIB. For a complete definition, see the IF-MIB.my file.

OID—MIB object identifier. An object identifier is expressed as a series of integers or text strings. Technically, the numeric form is the *object name* and the text form is the *object descriptor*. In practice, both are called object identifiers or OIDs. For example, the object name for the interfaces MIB is 1.3.6.1.2.1.2, and the object descriptor is ‘iso.internet.mgmt.mib-2.interfaces’, but either can be referred to as the OID. An OID can also be expressed as a combination of the two, such as iso.internet.2.1.2.