CHAPTER 6

Understanding SNMP

This chapter provides information about Cisco Access Registrar support for SNMP.

Overview

Cisco Access Registrar 1.7 includes enhancements that provide SNMP MIB and trap support for users of network management systems. The supported MIBs enable the network management station to collect state and statistic information from an Cisco AR server. The traps enable Cisco AR to notify interested network management stations of failure or impending failure conditions.

Cisco Access Registrar supports the MIBs defined in the following RFCs:

- RADIUS Authentication Client MIB, RFC 2618
- RADIUS Authentication Server MIB, RFC 2619
- RADIUS Accounting Client MIB, RFC 2620
- RADIUS Accounting Server MIB, RFC 2621

Cisco Access Registrar 1.7 MIB support enables a standard SNMP management station to check the current state of the server as well as the statistics on each client or each proxied remote server.

Cisco Access Registrar 1.7 Trap support enables a standard SNMP management station to receive trap messages from an Cisco AR server. These messages contain information indicating that either the server was brought up or down, or that the proxied remote server is down or has come back online.

Supported MIBs

The MIBs supported by Access Registrar enable a standard SNMP management station to check the current state of the server and statistics for each client or proxied remote server.

RADIUS-AUTH-CLIENT-MIB

The RADIUS-AUTH-CLIENT-MIB describes the client side of the RADIUS authentication protocol. The information contained in this MIB is useful when an Cisco AR server is used as a proxy server.

RADIUS-AUTH-SERVER-MIB

The RADIUS-AUTH-SERVER-MIB describes the server side of the RADIUS authentication protocol. The information contained in this MIB describes managed objects used for managing a RADIUS authentication server.

RADIUS-ACC-CLIENT-MIB

The RADIUS-ACC-CLIENT-MIB describes the client side of the RADIUS accounting protocol. The information contained in this MIB is useful when a Cisco AR server is used for accounting.

RADIUS-ACC-SERVER-MIB

The RADIUS-ACC-CLIENT-MIB describes the server side of the RADIUS accounting protocol. The information contained in this MIB is useful when a Cisco AR server is used for accounting.

SNMP Traps

The traps supported by Access Registrar enable a standard SNMP management station to receive trap messages from an Cisco AR server. These messages contain information indicating whether a server was brought up or down, or that the proxied remote server is down or has come back online.

A trap is a network message of a specific format issued by an SNMP entity on behalf of a network management agent application. A trap is used to provide the management station with an asynchronous notification of an event.

When a trap is generated, a single copy of the trap is transmitted as a trap PDU to each destination contained within a list of trap recipients.

The list of trap recipients is shared by all events and is determined at server initialization time along with other trap configuration information. The list of trap recipients dictates where Cisco AR traps are directed.

The configuration of any other SNMP agent on the host is ignored. By default, all traps are enabled but no trap recipients are defined. By default, no trap is sent until trap recipients are defined.

For detailed information about how to configure traps, refer to the *Configuring Cisco Access Registrar* chapter in the *Cisco Access Registrar Installation and Configuration Guide*. To configure SNMP traps, do the following:

1. Stop the Cisco AR server (**etc/init.d/arservagt stop**).
2. Modify the **/cisco-ar/ucd-snmp/share/snmp/snmpd.conf** file.
3. Restart the Cisco AR server (**etc/init.d/arservagt restart**).

When you configure traps, you must provide the following information:

- List of trap recipients (community string for each)
- Suppressing traps for any type of message
- Frequency of traps for any type of message

Supported Traps

The traps supported by Cisco Access Registrar enable Cisco AR to notify interested management stations of events, failure, or impending failure conditions. Traps are a network message of a specific format issued by an SNMP entity on behalf of a network management agent application. Traps are used to provide the management station with an asynchronous notification of an event.

carServerStart

carServerStart signifies that the server has started on the host from which this notification was sent. This trap has one object, *carNotifStartType,* which indicates the start type. A *firstStart* indicates this is the server process' first start. *reload* indicates this server process has an internal reload. This typically occurs after rereading some configuration changes, but *reload* indicates this server process did not quit during the reload process.

carServerStop

carServerStop signifies that the server has stopped normally on the host from which this notification was sent.

carInputQueueFull

carInputQueueFull indicates that the percentage of use of the packet input queue has reached its high threshold. This trap has two objects:

- *carNotifInputQueueHighThreshold*—indicates the high limit percentage of input queue usage
- *carNotifInputQueueLowThreshold*—indicates the low limit percentage of input queue usage

By default, *carNotifInputQueueHighThreshold* is set to 90% and *carNotifInputQueueLowThreshold* is set to 60%.

Note The values for these objects cannot be changed at this time. You will be able to modify them in a future release of Cisco Access Registrar.

After this notification has been sent, another notification of this type will not be sent again until the percentage usage of the input queue goes below the low threshold.

If the percentage usage reaches 100%, successive requests may be dropped, and the server may stop responding to client requests until the queue drops down again.

carInputQueueNotVeryFull

carInputQueueNotVeryFull indicates that the percentage usage of the packet input queue has dropped below the low threshold defined in *carNotifInputQueueLowThreshold*. This trap has two objects:

- *carNotifInputQueueHighThreshold*—indicates the high limit percentage of input queue usage
- *carNotifInputQueueLowThreshold*—indicates the low limit percentage of input queue usage

After this type of notification has been sent, it will not be sent again until the percentage usage goes back up above the high threshold defined in *carNotifInputQueueHighThreshold*.

carOtherAuthServerNotResponding

carOtherAuthServerNotResponding indicates that an authentication server is not responding to a request sent from this server. This trap has three objects:

- *radiusAuthServerAddress*—indicates the identity of the concerned server
- *radiusAuthClientServerPortNumber*—indicates the port number of the concerned server
- *carAuthServerType*—indicates the type of the concerned server

The index of these three objects identifies the entry in *radiusAuthServerTable* and *carAccServerExtTable* which maintains the characteristics of the concerned server.

Note One should not rely solely on **carOtherAuthServerNotResponding** for server state. Several conditions, including a restart of the Cisco AR server, could result in either multiple *carOtherAuthServerNotResponding* notifications being sent or in a *carOtherAuthServerResponding* notification *not* being sent. NMS can query the *carAuthServerRunningState* in *carAuthServerExtTable* for the current running state of this server.

carOtherAuthServerResponding

carOtherAuthServerResponding signifies that an authentication server which had formerly been in a *down* state is now responding to requests from the Cisco AR server. This trap has three objects:

- *radiusAuthServerAddress*—indicates the identity of the concerned server
- *radiusAuthClientServerPortNumber*—indicates the port number of the concerned server
- *carAuthServerType*—indicates the type of the concerned server

The index of these three objects identifies the entry in *radiusAuthServerTable* and *carAccServerExtTable* which maintains the characteristics of the concerned server.

One should not rely on receiving this notification as an indication that all is well with the network. Several conditions, including a restart of the Cisco AR server, could result in either multiple *carOtherAuthServerNotResponding* notifications being sent or in a *carOtherAuthServerResponding* notification *not* being sent. The NMS can query the *carAuthServerRunningState* in *carAuthServerExtTable* for the current running state of this server.

carOtherAccServerNotResponding

carOtherAuthServerNotResponding signifies that an accounting server is not responding to the requests sent from this server. This trap has three objects:

- *radiusAccServerAddress*—indicates the identity of the concerned server
- *radiusAccClientServerPortNumber*—indicates the port number of the concerned server
- *carAcchServerType*—indicates the type of the concerned server

The index of these three objects identifies the entry in *radiusAuthServerTable* and *arAccServerExtTable* which maintains the characteristics of the concerned server.

One should not solely rely on this for server state. Several conditions, including the restart of the Cisco AR server, could result in either multiple *carOtherAccServerNotResponding* notifications being sent or in a *carOtherAccServerResponding* notification *not* being sent. The NMS can query the *carAccServerRunningState* in *carAccServerExtTable* for current running state of this server.

carOtherAccServerResponding

carOtherAccServerResponding signifies that an accounting server that had previously sent a *not responding* message is now responding to requests from the Cisco AR server. This trap has three objects:

- *radiusAccServerAddress*—indicates the identity of the concerned server
- *radiusAccClientServerPortNumber*—indicates the port number of the concerned server
- *carAccServerType*—indicates the type of the concerned server

The index of these three objects identifies the entry in *radiusAuthServerTable* and *arAccServerExtTable* which maintains the characteristics of the concerned server.

One should not rely on the reception of this notification as an indication that all is well with the network. Several conditions, including the restart of the Cisco AR server, could result in either multiple *carOtherAccServerNotResponding* notifications being sent or in a **carOtherAccServerResponding** notification *not* being sent. The NMS can query the *carAccServerRunningState* in *carAccServerExtTable* for the current running state of this server.

carAccountingLoggingFailure

carAccountingLoggingFailure signifies that this Cisco AR server cannot record accounting packets locally. This trap has two objects:

- *carNotifAcctLogErrorReason*—indicates the reason packets cannot be recorded locally
- *carNotifAcctLogErrorInterval*—indicates how long to wait until another notification of this type might be sent. A value of 0 (zero) indicates no time interval checking, meaning that no new notification can be sent until the error condition is corrected.

Configuring Traps

Cisco Access Registrar's SNMP implementation uses various configuration files to configure its applications.

Directories Searched

Configuration files can be found and read from numerous places. By default, SNMP applications look for configuration files in the following three directories (in the order listed):

1. **/usr/local/share/snmp/snmp.conf**

   This directory contains common configuration for the agent and the application. Refer to man page **snmp.conf(5)** for details.

2. **/usr/local/share/snmp/snmpd.conf**
3. **/usr/local/share/snmp/snmp.local.conf**

   This directory configures the agent. Refer to man page **snmp.conf(5)** for details.

In each of these directories, an SNMP application looks for files with the extension ***.conf***. The application also looks for configuration files in default locations where a configuration file can exist for any given configuration file type.

These files are optional and are only used to configure the extensible portions of the agent, the values of the community strings, and the optional trap destinations. By default, the first community string ("public" by default) is allowed read-only access and the second ("private" by default) is allowed write access, as well. The third to fifth community strings are also read-only.

Additionally, the above default search path can be over-ridden by setting the environmental variable SNMPCONFPATH to a colon-separated list of directories to search.

Finally, applications that store persistent data will also look for configuration files in the **/var/snmp** directory.

Configuration File Types

Each application may use multiple configuration files which will configure various different aspects of the application. For instance, the SNMP agent (**snmpd**) knows how to understand configuration directives in both the **snmpd.conf** and the **snmp.conf** files. In fact, most applications understand how to read the contents of the **snmp.conf** files. Note, however, that configuration directives understood in one file may not be understood in another file. For further information, read the associated manual page with each configuration file type. Also, most of the applications support a '-H' switch on the command line that will list the configuration files it will look for and the directives in each one that it understands.

The **snmp.conf** configuration file is intended to be a application suite-wide configuration file that supports directives that are useful for controlling the fundamental nature of all of the SNMP applications, such as how they all manipulate and parse the textual SNMP MIB files.

Switching Configuration Files in Mid-File

It's possible to switch in mid-file the configuration type that the parser is supposed to be reading. Since that output for the agent by default, but you didn't want to do that for the rest of the applications (for example, **snmpget** and **snmpwalk**, you would need to put a line like the following into the **snmp.conf** file.

```
dumpPacket true
```

But, this would turn it on for all of the applications. So, instead, you can put the same line in the snmpd.conf file so that it only applies to the snmpd demon. However, you need to tell the parser to expect this line. You do this by putting a special type specification token inside a square bracket ([ ]) set. In other words, inside your **snmpd.conf** file you could put the above **snmp.conf** directive by adding a line like the following:

```
[snmp] dumpPacket true
```

This tells the parser to parse the above line as if it were inside a **snmp.conf** file instead of an snmpd.conf file. If you want to parse a bunch of lines rather than just one then you can make the context switch apply to the remainder of the file or until the next context switch directive by putting the special token on a line by itself:

```
# make this file handle snmp.conf tokens:
[snmp]
dumpPacket true
logTimestamp true
# return to our original snmpd.conf tokens:
[snmpd]
rocommunity mypublic
```

Community String

A community string is used to authenticate the trap message sender (SNMP agent) to the trap recipient (SNMP management station). A community string is required in the list of trap receivers.