



Configuring Remote Monitoring (RMON)

This chapter provides configuration and viewing information for Remote Monitoring (RMON). Information in this chapter applies to all CSS models except where noted.

This chapter contains the following sections:

- RMON Overview
- RMON Configuration Considerations
- Configuring an RMON Event
- Configuring an RMON Alarm
- Configuring an RMON History
- Viewing RMON Information
- RMON Configuration in a Startup-Config File

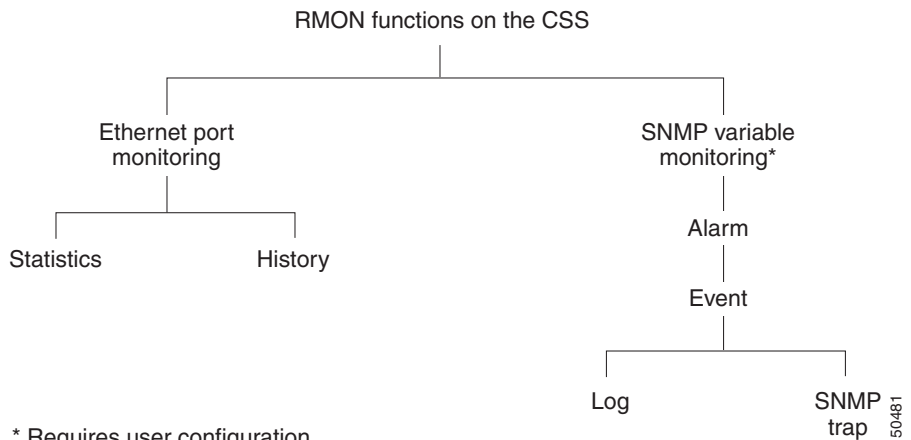
RMON Overview

RMON allows you to remotely monitor and analyze the activity of packets on CSS Ethernet ports. It also allows alarm configuration for monitoring MIB objects, and the event configuration to notify you of these alarm conditions. For detailed information about RMON and its MIB objects, refer to RFC 1757.

The version of RMON provided on the CSS is a subset of the RMON-1 groups. The CSS supports the following groups:

- **Group 1 - (Statistics)** Provides data about all Ethernet ports on a CSS. You cannot configure RMON statistics. You can only view them.
- **Group 2 - (History)** Provides data about the Ethernet ports over a historical period. Histories are preconfigured for each port. You can configure additional port histories.
- **Group 3 - (Alarm)** Allows you to create an alarm and configure the conditions, based on a MIB object, to trigger an alarm when significant events occur.
- **Group 9 - (Event)** Allows you to create an event and configure the event action when its associated alarm occurs.

Figure 10-1 Supported RMON Functions on the CSS



RMON Configuration Considerations

Consider the following before you implement RMON functionality on your CSS:

- You can configure an RMON event, alarm, and history. You cannot configure any CSS attributes for RMON statistics. Statistics for the ports are only viewable by using the **show rmon** command.
- You cannot change the configuration for an RMON event, alarm, or history after you activate it. If you need to change a configuration after activation, you must delete it first and then recreate it with the necessary changes. Note that you can change your configuration at any time before you activate it.
- You must assign an RMON event to an alarm. Thus, you must create the event before you can configure it to an alarm.
- RMON histories are preconfigured for each Ethernet port. Though these histories cannot be deleted or modified, you can add additional history entries for a port. For more information on the preconfigured histories and adding more history entries, refer to “Configuring an RMON History” later in this chapter.

The sections in this chapter for configuring events, alarms, and histories provide quick configuration tables. If you need additional configuration information, refer to the sections that follow the tables.

Configuring an RMON Event

An RMON event is the action that occurs when an associated RMON alarm is triggered. When an alarm event occurs, it can be configured to generate a log event, a trap to an SNMP network management station, or both. For information on viewing alarm events in log files, refer to “Viewing Events in a Log File” later in this chapter. For more information on configuring SNMP on your CSS, refer to Chapter 9, Configuring Simple Network Management Protocol (SNMP).

If you are familiar with configuring the attributes for an RMON event, refer to Table 10-1. The table contains the steps to configure an event, their possible settings, and an example for each step. For more detailed information on configuring an event, refer to the sections after the following table.

Table 10-1 RMON Event Configuration Quick Start

Steps and Possible Settings

1. From global configuration mode, create an RMON event configuration identifier. Enter a number from 1 to 65534.

```
(config)# rmon-event 1
```

2. Assign an existing SNMP community for this event. Enter the community name configured by using the (config) **snmp community** command. This step is only required if the traps are sent to an SNMP network management station.

```
(config-rmonevent[1])# community moonbase_alpha
```

3. Provide a description for the event. Enter a quoted string with a maximum of 126 characters including spaces.

```
(config-rmonevent[1])# description "This event occurs when service connections exceed 100"
```

4. Assign the owner who defined and is using the resources of the event. Enter a quoted string with a maximum of 126 characters including spaces. You must define the owner before you can activate the event.

```
(config-rmonevent[1])# owner "Boston Tech Lab"
```

5. Specify the type of event notification. The type determines where the notification is sent. The options are **type log**, **type trap**, or **type log-and-trap**.

```
(config-rmonevent[1])# type log-and-trap
```

6. Activate the event.

```
(config-rmonevent[1])# active
```

For information on configuring an alarm and associating this event to an alarm, refer to “Configuring an RMON Alarm” later in this chapter.

Creating a Configuration Identifier for an RMON Event

The RMON event configuration identifier identifies the event to the CSS. This allows you to assign specific configuration attributes to the identifier. When you create an identifier, you access the configuration mode for that event automatically.

To create an event configuration identifier, use the **rmon-event** *index* command from any configuration mode except boot and RMON alarm configuration modes. The *index* is a number from 1 to 65535.



Note

The RMON event index 65535 is administratively predefined and cannot be modified. If you enter this index number, a message similar to the following appears: %% Index internally used. Administrative control not allowed.

For example, to create a RMON event with an identifier of 1, access global configuration mode and enter:

```
(config)# rmon-event 1
```

To see a list of existing RMON event configuration identifiers, enter:

```
(config)# rmon-event ?
```

After you create the identifier for the event, the prompt changes to (config-rmonevent[1]). Now you can define the event, as described in “Setting the RMON Event Attributes” later in this chapter.

Modifying the Attributes for an Existing RMON Event Configuration Identifier

When you have created an RMON event configuration identifier but you have not activated it, you can modify its attributes.

**Note**

If the event configuration identifier is activated, you cannot modify its attributes. You must delete it, recreate it, and respecify its attributes.

To modify the attributes, you must access the RMON event configuration mode for that event. To access this mode from any configuration mode except boot and RMON alarm configuration modes, use the **rmon-event** command.

For example, to access the mode for RMON event 1, access global configuration mode and enter:

```
(config)# rmon-event 1
```

To see a list of existing RMON events, enter:

```
(config)# rmon-event ?
```

To modify the attributes, refer to “Setting the RMON Event Attributes” later in this chapter.

Deleting an RMON Event Configuration Identifier

If you have an active RMON event configuration identifier that requires changes to its attributes or you no longer need it, delete it. Before you delete an event identifier that requires changes, note the settings for its attributes.

To delete the event configuration identifier, use the **no rmon-event** command. For example, to delete RMON event 1 and its configuration, access global configuration mode and enter:

```
(config)# no rmon-event 1
```

After you delete the identifier to change its attributes, recreate it as described in “Creating a Configuration Identifier for an RMON Event” later in this chapter.

Setting the RMON Event Attributes

After you create an RMON event identifier or access RMON event configuration mode for an existing inactive event identifier, you can set its attributes as described in the following sections:

- Defining an Event Community
- Describing an Event
- Assigning an Owner
- Defining the Notification of an Event

After you set the attributes, activate the event as described in “Activating the Event” later in this chapter.

Defining an Event Community

When an alarm event occurs and the event is configured to send an SNMP trap, the CSS sends the trap to the trap host with the specified community. To define a community to an unactivated event, use the **community** *community_name* command. The *community_name* variable is the name of the SNMP community you configured using the **snmp community** command (refer to “Configuring an SNMP Community” in Chapter 9, Configuring Simple Network Management Protocol (SNMP)).

For example, to define the SNMP moonbase_alpha community for this event, enter:

```
(config-rmonevent[1])# community moonbase_alpha
```

Describing an Event

When an alarm event occurs, the CSS sends a description with the event notification. Because a description is not generated automatically, you must provide one. To provide a description, use the **description** “*description*” command. The *description* variable is the description for the RMON event. Enter a quoted text string with a maximum length of 126 characters.

For example, to provide a description for the event, enter:

```
(config-rmonevent[1])# description "This event occurs when service connections exceed 100"
```

Assigning an Owner

You must define the entity that configured this RMON event and is using the resources assigned to it. To define the owner, use the **owner** “*owner_name*” command. The *owner_name* variable is a quoted text string with a maximum of 126 characters. The owner for the event must be the same as the owner for the alarm.

For example, to define the owner named Boston Tech Lab, enter:

```
(config-rmonevent[1])# owner "Boston Tech Lab"
```

Defining the Notification of an Event

When an RMON event occurs, the event type determines where the CSS sends the event notification.

- A log event type designates that the event notification is made in a CSS log location (for example, CSS disk log file or session). For information on viewing log files, refer to “Viewing Events in a Log File” later in this chapter.

To define the event as a log type, enter:

```
(config-rmonevent[1])# type log
```

- A trap event type designates that a trap is sent to a SNMP network management station. To define the event as a trap type, enter:

```
(config-rmonevent[1])# type trap
```



Note

When you want the event to send a trap to a network management station, you need to configure SNMP. For more information on SNMP, refer to Chapter 9, Configuring Simple Network Management Protocol (SNMP).

- You can also designate that the event type is both log and trap. To define the event as both log and trap types, enter:

```
(config-rmonevent[1])# type log-and-trap
```

Activating the Event

After you configure the event attributes, you can activate the event. Before you can activate an event, you must specify the owner of the event as described in “Assigning an Owner” earlier in this chapter. To activate the event, enter:

```
(config-rmonevent [1]) # active
```

**Note**

Before you activate the event, make sure that you are finished configuring it and are satisfied with its settings. After you activate an event, you cannot modify its configuration settings. The only way to change the event is to delete it, and then recreate it.

Configuring an RMON Alarm

An RMON alarm allows you to monitor a MIB object for a desired transitory state. An alarm periodically takes samples of the object’s value and compares them to the configured thresholds.

RMON allows you to configure two types of sampling, absolute and delta:

- Absolute sampling compares the sample value directly to the threshold. This sampling is similar to a gauge, recording values that go up or down.
- Delta sampling subtracts the current sample value from the last sample taken, and then compares the difference to the threshold. This sampling is similar to a counter, recording a value that is constantly increasing.

When a sample value crosses an alarm threshold, an associated event is generated. To limit the number of generated events, only one event is generated when a threshold is crossed. The CSS does not generate additional events until an opposite threshold is crossed. For example, when a rising threshold is crossed, one event is generated. The next event occurs only when a falling threshold is crossed.

When you associate an event to an alarm and an alarm occurs, the event defines the action the CSS takes when an alarm occurs. For more information on events, refer to “Configuring an RMON Event” earlier in this chapter.

Figure 10-2 is an example of absolute sampling.

Figure 10-2 Example of Absolute Sampling

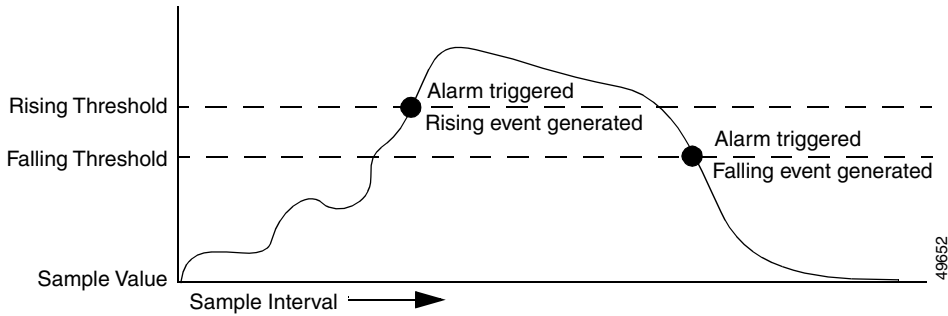
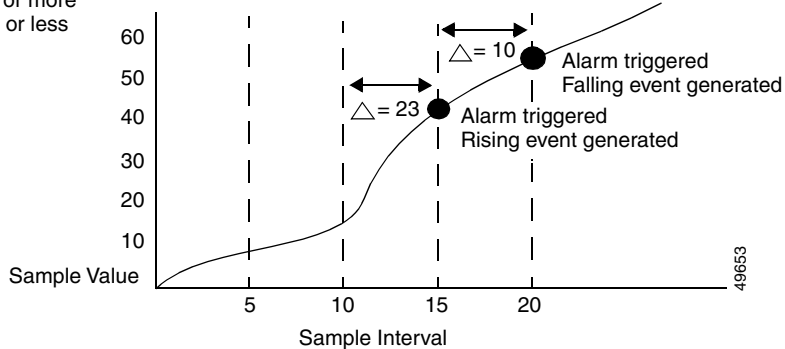


Figure 10-3 is an example of delta sampling.

Figure 10-3 Example of Delta Sampling

Rising Threshold = 20 or more
Falling Threshold = 10 or less



RMON Alarm Configuration Quick Start

If you are familiar with configuring the attributes for an RMON alarm, refer to Table 10-2. The table contains the steps to configure the alarm, its possible settings, and an example for each step. For more detailed information on configuring an alarm, refer to the sections following Table 10-2.

Table 10-2 RMON Alarm Configuration Quick Start

Steps and Possible Settings

1. From global configuration mode, create a configuration identifier for an RMON alarm. Enter a number from 1 to 65534.

```
(config)# rmon-alarm 1
```
 2. Assign the owner who defined and is using the resources of the alarm. Enter a quoted string with a maximum of 32 characters including spaces. The owner must be the same as the owner for the event.

```
(config-rmonalarm[1])# owner "Boston Tech Lab"
```
 3. Define the MIB object for the sample variable. For example, for the current number of connections for this service, enter `apSvcConnections`. To see a list of objects, use the **sample-variable ?** command. For detailed information about an object, use the **lookup** command.

```
(config-rmonalarm[1])# sample-variable apSvcConnections
```
 4. Define the sampling type. The options are **absolute** or **delta**.

```
(config-rmonalarm[1])# sample-type absolute
```
 5. Define the startup alarm type. The options are **falling**, **rising**, or **rising-and-falling**.

```
(config-rmonalarm[1])# startup-type rising-and-falling
```
 6. Define the rising threshold. Enter a number from 0 to 4294967295.

```
(config-rmonalarm[1])# rising-threshold 100
```
 7. Associate the rising threshold with an existing RMON event. Enter a number from 0 to 65535. If you enter 0, no event is generated.

```
(config-rmonalarm[1])# rising-event 1
```
 8. Define the falling threshold. Enter a number from 0 to 4294967295.

```
(config-rmonalarm[1])# falling-threshold 90
```
-

Table 10-2 RMON Alarm Configuration Quick Start (continued)**Steps and Possible Settings**

9. Associate the falling threshold with an existing RMON event. Enter a number from 0 to 65535. If you enter 0, no event is generated.

```
(config-rmonalarm[1])# falling-event 2
```

10. Specify the sampling interval for the RMON alarm. The interval is in seconds. Enter a number from 1 to 65535.

```
(config-rmonalarm[1])# sample-interval 30
```

11. Activate the alarm.

```
(config-rmonalarm[1])# active
```

Creating a Configuration Identifier for an RMON Alarm

The RMON alarm configuration identifier identifies the alarm to the CSS. This allows you to assign specific configuration attributes to the identifier. When you create an identifier, you access the configuration mode for that alarm automatically.

To create an alarm configuration identifier, use the **rmon-alarm *index*** command from any configuration mode except boot and RMON history configuration modes. The *index* is a number from 1 to 65535.

**Note**

The RMON alarm index 65535 is administratively predefined and cannot be modified. If you enter this index number, a message similar to the following appears: %% Index internally used. Administrative control not allowed.

For example, to create an RMON alarm with an identifier of 1, access global configuration mode and enter:

```
(config)# rmon-alarm 1
```

To see a list of existing RMON alarm configuration identifiers, enter **rmon-alarm ?**.

After you create the identifier for the alarm, the prompt changes to `(config-rmonalarm[1])`. Now you can define the alarm, as described in “Setting the RMON Alarm Attributes” later in this chapter.

Modifying Attributes for an Existing RMON Alarm Configuration Identifier

When you have already created an RMON alarm configuration identifier and its attributes but you have not activated it, you can modify its attributes.



Note

If the alarm configuration is activated, you cannot modify its settings. You must delete the alarm configuration, recreate it, and respecify its attributes.

To modify the attributes, you must access the RMON alarm configuration mode for that alarm. To access this mode from any configuration mode except boot and RMON history configuration modes, use the **rmon-alarm** command. For example, to access the mode for RMON alarm 1, access global configuration mode and enter:

```
(config)# rmon-alarm 1
```

To see a list of existing RMON alarms, enter:

```
(config)# rmon-alarm ?
```

To modify the attributes, refer to “Setting the RMON Alarm Attributes” later in this chapter.

Deleting an RMON Alarm Configuration Identifier

If you have an active RMON alarm configuration identifier that requires changes to its attributes or you no longer need an alarm identifier, delete it. Before you delete the configuration that requires changes, note the settings for its attributes.

To delete the alarm configuration identifier, use the **no rmon-alarm** command. For example, to delete RMON alarm 1 and its configuration, access global configuration mode and enter:

```
(config)# no rmon-alarm 1
```

After you delete the alarm identifier to change its attributes, recreate the identifier as described in “Creating a Configuration Identifier for an RMON Alarm” earlier in this chapter.

Setting the RMON Alarm Attributes

After you create an RMON alarm identifier or access RMON alarm configuration mode for an existing inactive alarm identifier, you can set its attributes as described in the following sections:

- Assigning an Owner
- Finding and Defining a Sample Variable
- Defining an Absolute or Delta Sampling
- Defining a Rising Threshold and Index
- Defining a Falling Threshold and Index
- Defining a Startup Alarm
- Defining the Sampling Interval

After you set all of the attributes, activate the alarm as described in “Activating an Alarm” later in this chapter.

Assigning an Owner

You must define the entity that configured the RMON alarm and is using the resources assigned to it. To define the owner, use the **owner** “*owner_name*” command. The *owner_name* variable is a quoted text string with a maximum of 32 characters. Enter the same name as the owner of the event.

For example, to define the owner named Boston Tech Lab, enter:

```
(config-rmonalarm[1])# owner "Boston Tech Lab"
```

Finding and Defining a Sample Variable

For an alarm condition, RMON samples a configured sample variable associated with a MIB object. MIB objects to consider include the following:

- `svcExt.mib` contains service objects (for example, `apSvcConnections` is the MIB object for the current number of TCP connections to this service).
- `cntExt.mib` contains content rule objects (for example, `apCntHits` is the MIB object for the total number of hits on this service for this content rule).



Note

For more information on CSS MIBs, refer to Chapter 9, Configuring Simple Network Management Protocol (SNMP).

To look up a MIB object and view its description, use the **lookup** command. For example, to view the description for the `apSvcConnections` object, enter:

```
(config-rmonalarm[1])# lookup apSvcConnections
ASN Name:          apSvcConnections
MIB:               svcext
Object Identifier: 1.3.6.1.4.1.2467.1.15.2.1.20
Argument Type:    Integer
Range:            0-4294967295
Description:
    The current number of TCP connections to this service
```

To specify the sample variable for this RMON alarm, use the **sample-variable mib_object** command. For example, to define the `apSvcConnections` MIB object for the current number of service connections, enter:

```
(config-rmonalarm[1])# sample-variable apSvcConnections
```

To see a list of SNMP variables, use the **sample-variable ?** command. For example:

```
(config-rmonalarm[1])# sample-variable ?

apSvcLoadInfoTimeout
apSvcLoadSvcStatRptTimeout
apSvcLoadEnable
apSvcLoadDecayInterval
apSvcLoadStepStatic
apSvcLoadStepSize
apSvcLoadThreshold
...
```

Defining an Absolute or Delta Sampling

When you configure an alarm, you can define the sampling method to compare the sample value of a MIB object to either:

- The configured threshold directly. This sampling is like a gauge, recording the value as it goes up and down. Refer to Figure 10-1.
- The previous sampling, and then their difference is compared to the configured threshold. This sampling is like a counter, recording the value that is constantly increasing. Refer to Figure 10-2.

Absolute sampling compares the sample value to the configured threshold. For example, if you want to know when 30,000 service connections occur on the CSS during a sample interval, configure the `apSvcConnections` MIB object with absolute sampling. The `apSvcConnections` object is the current number of connections on a service. To define an absolute sampling, enter:

```
(config-rmonalarm[1])# sample-type absolute
```

Delta sampling compares the current sample value with the previous sample and compares their difference to the configured threshold. For example, if you want to know when the number of content rule hits increase by 100,000 hits compared to its previous sampling, configure the `apCntHits` MIB object with delta sampling. `apCntHits` is an ever-increasing count of hits. To define a delta sampling, enter:

```
(config-rmonalarm[1])# sample-type delta
```

Defining a Rising Threshold and Index

When you want to be notified when a sampling is greater than or equal to a specific number, set a rising threshold and associate it to a configured event.



Note

You must create an RMON event before you can associate it with an alarm.

For a single rising alarm event to occur, a sampled value is greater than or equal to the rising threshold value, and the value at the last sampling interval is less than this threshold.

- To set the threshold for the alarm, use the **rising-threshold** *rising_value* command. The *rising_value* variable is the threshold for the rising sample type. Enter an integer from 0 to 4294967295.

For example, to set the rising threshold value of 100, enter:

```
(config-rmonalarm[1])# rising-threshold 100
```

- To associate a configured event to the RMON alarm when the rising threshold is exceeded, use the **rising-event** *rising_index* command. The *rising_index* variable is the event index used when a rising threshold is crossed. If you enter 0, no event is generated.

For example, to associate the threshold to RMON event 1, enter:

```
(config-rmonalarm[1])# rising-event 1
```

To see a list of RMON events, enter:

```
(config-rmonalarm[1])# rising-event ?
```

Defining a Falling Threshold and Index

When you want to be notified when a sampling is less or equal to a specific number, set a falling threshold and associate it to a configured event.



Note

You must create an RMON event before you can associate it with an alarm.

For a single falling alarm event to occur, a sampled value is less than or equal to the falling threshold value, and the value at the last sampling interval is greater than this threshold.

- To set the threshold for the alarm, use the **falling-threshold** *falling_value* command. The *falling_value* variable is the threshold for the falling sample type. Enter an integer from 0 to 4294967295.

For example, to set the falling threshold value of 90, enter:

```
(config-rmonalarm[1])# falling-threshold 90
```

- To associate a configured event to the RMON alarm when the falling threshold is exceeded, use the **falling-event** *falling_index* command. The *falling_index* variable is the event index used when a falling threshold is crossed. If you enter 0, no event is generated.

For example, to associate the threshold to RMON event 2, enter:

```
(config-rmonalarm[1])# falling-event 2
```

To see a list of RMON events, enter:

```
(config-rmonalarm[1])# falling-event ?
```

Defining a Startup Alarm

A startup alarm allows the CSS to generate an alarm when the first sample triggers a falling or rising threshold.

- A startup falling alarm occurs when the first sample is less than or equal to the falling threshold. To enable this alarm, enter:

```
(config-rmonalarm[1])# startup-type falling
```

- A startup rising alarm occurs when the first sample is greater than or equal to the rising threshold. To enable this alarm, enter:

```
(config-rmonalarm[1])# startup-type rising
```

- To enable an alarm when either a falling or rising threshold is triggered, enter:

```
(config-rmonalarm[1])# startup-type rising-and-falling
```

Defining the Sampling Interval

The sampling interval is the time interval, in seconds, over which the data is sampled and compared with the rising and falling thresholds. To specify the sampling interval for this RMON alarm, use the **sample-interval** *interval* command. The *interval* variable is the number of seconds, from 1 to 65535.

For example, to enter a sampling interval of 60 seconds, enter:

```
(config-rmonalarm[1])# sample-interval 60
```

With delta sampling, set the sampling interval short enough so that the sampled variable, which has a tendency to go up and down very fast, does not wrap during a single sampling period.

Activating an Alarm

After you configure the alarm attributes, you can activate the alarm. Before you can activate an alarm, you must specify all attributes for the alarm. To activate the alarm, enter:

```
(config-rmonalarm[1])# active
```

**Note**

Before you activate the alarm, make sure that you are finished configuring it and are satisfied with its settings. After you activate an alarm, you cannot modify its configuration settings. The only way to change the alarm is to delete it, and then recreate it.

Configuring an RMON History

You can configure the operation of the RMON history that periodically samples any CSS Ethernet port for statistical data. All ports are preconfigured with histories for 30-second and 30-minute intervals, and 50 buckets with one sample per bucket. However, you can create additional histories for a specific port. This allows you to configure the time interval to take the sample and the number of samples you want to save.

You can view the statistical information for the history by using the **show rmon-history** command. For more information about viewing the history, refer to “Viewing History” later in this chapter.

If you are familiar with configuring the attributes for an RMON history, refer to Table 10-3. The table contains the steps to configure the history, their possible settings, and an example for each step. For more detailed information on configuring a history, refer to the sections following Table 10-3.

Table 10-3 RMON History Configuration Quick Start**Steps and Possible Settings**

1. From global configuration mode, create an RMON history. Enter a number from 1 to 65535.

```
(config)# rmon-history 5
```

2. Assign the owner who defined and is using the history resources. Enter up to 32 characters.

```
(config-rmonhistory[5])# owner Boston_Tech_Lab
```

3. Define the data source object for the Ethernet port. The port is identified by an ifIndex data object identifier. To see a list of data object IDs, use the **show interface** command.

```
(config-rmonhistory[5])# data-source ifIndex.3
```

4. Define the time interval for the history. The interval is in seconds. Enter a number from 1 to 3600. The default is 1800.

```
(config-rmonhistory[5])# interval 60
```

5. Define the bucket count for the interval. Enter a number from 1 to 65535. The default is 50.

```
(config-rmonhistory[5])# requested-buckets 25
```

6. Activate the history.

```
(config-rmonhistory[5])# active
```

Creating a Configuration Identifier for an RMON History

The RMON history configuration identifier identifies the history to the CSS. This allows you to assign specific configuration attributes to the identifier. When you create an identifier, you access the configuration mode for that history automatically.

To create a history identifier, use the **rmon-history index** command from any configuration mode except boot configuration mode. The *index* variable is the index number that identifies the history. Enter an integer from 1 to 65535.

For example, to create an RMON history identifier 5, access global configuration mode and enter:

```
(config)# rmon-history 5
```

**Note**

Some history index numbers are administratively predefined and cannot be modified. If you enter an index number under administrative control, a message similar to the following appears:

```
%% Index internally used. Administrative control not allowed.
```

After you create the identifier, the prompt changes to `(config-rmonhistory [5])`. Now you can define the history, as described in “Setting the RMON History Attributes” later in this chapter.

Modifying the Attributes for an Existing RMON History Configuration Identifier

When you have already created an RMON history identifier but you have not activated it, you can modify its attributes.

**Note**

If the history is activated, you cannot modify its settings. You must delete the history, recreate it, and respecify its attributes.

To modify the attributes, you must access the RMON history configuration mode for that history. To access this mode from any configuration mode except boot configuration mode, use the **rmon-history** command. For example, to access the mode for RMON history 5, access global configuration mode and enter:

```
(config)# rmon-history 5
```

To see a list of existing RMON histories, enter:

```
(config)# rmon-history ?
```

**Note**

Some history index numbers are administratively predefined and cannot be modified. If you enter an index number under administrative control, a message similar to the following appears:

```
%% Index internally used. Administrative control not allowed.
```

To modify the history attributes, refer to the “Setting the RMON History Attributes” later in this chapter.

Deleting an RMON History Configuration Identifier

If you have an active RMON history configuration identifier that requires changes to its attributes or you no longer need the identifier, delete it. Before you delete the identifier that requires changes, note the settings for its attributes.

To delete an RMON history configuration identifier, use the **no rmon-history** command. For example, to delete RMON history 5, access global configuration mode and enter:

```
(config)# no rmon-history 5
```

After you delete the history identifier to change its attributes, recreate it as described in “Creating a Configuration Identifier for an RMON History” later in this chapter.

Setting the RMON History Attributes

After you create an RMON history or access RMON history configuration mode for an existing inactive alarm, you can set its attributes as described in the following sections:

- Defining the Data Object
- Assigning an Owner
- Defining the Bucket Count
- Defining the Bucket Interval

After you set the attributes, activate the history as described in “Activating an RMON History Entry” later in this chapter.

Defining the Data Object

When you create a history, you must associate it with a CSS Fast Ethernet or Gigabit Ethernet port. To define the data object, use the **data-source** *port* command. The *port* is identified by an ifIndex data object identifier. For example, if your CSS has 12 Ethernet ports, they have data object IDs of ifIndex.1 through ifIndex.12. The Ethernet management port has an ID of ifIndex.14.

For example, to define Ethernet port 4, enter:

```
(config-rmonhistory[5])# data-source ifIndex.4
```

To see a list of data object IDs for all of the CSS Ethernet ports, enter:

```
(config-rmonhistory[5])# show interface
```

Assigning an Owner

You must define the entity that configured the RMON history and is using the resources assigned to it. To define the owner, use the **owner** *owner_name* command. The *owner_name* variable is an unquoted text string with a maximum of 32 characters.

For example, to define an owner named Boston Tech Lab, enter:

```
(config-rmonhistory[5])# owner Boston_Tech_Lab
```

Defining the Bucket Count

You can define a bucket count which is the number of discrete sampling intervals over which data is saved for a history entry. To define a bucket count, use the **requested-buckets** *count* command. The *count* variable is an integer from 1 to 65535. The default is 50.

For example, to define a bucket count of 25, enter:

```
(config-rmonhistory[5])# requested-buckets 25
```

Defining the Bucket Interval

You can specify the time interval, in seconds, to take a bucket sample for an RMON history operation. To set this interval, use the **interval value** command. The *value* variable is the interval in seconds. Enter an integer from 1 to 3600. The default is 1800 (30 minutes).

For example, to define a time interval of 60 seconds, enter:

```
(config-rmonhistory[5])# interval 60
```

Activating an RMON History Entry

After you configure the history attributes, you can activate the history for the port. To activate an RMON history entry, use the **active** command.

**Note**

Before activating this command, you must specify the owner for the RMON history entry.

To activate the history, enter:

```
(config-rmonhistory[5])# active
```

**Note**

Before you activate the history, make sure that you are finished configuring it and are satisfied with its settings. After you activate a history, you cannot modify its configuration settings. The only way to change the history is to delete it, and then recreate it.

Viewing RMON Information

RMON information includes:

- Ethernet port statistics and history data that you can view from the CSS through **show** commands.
- Alarm events notifications that are sent to log locations on the CSS or an SNMP network management station. For information on configuring SNMP on the CSS, refer to Chapter 9, Configuring Simple Network Management Protocol (SNMP).

The following sections provide information on:

- Viewing Statistics
- Viewing History
- Viewing Events in a Log File

Viewing Statistics

RMON statistics provide a summary of data received in the Fast Ethernet or Gigabit Ethernet ports. You can view them either in a CSS CLI session through the **show rmon** command or directly through an SNMP network management station by using ether-stats MIB objects (refer to RFC1398).

The CSS **show rmon** command allows you to display the extended 64-bit RMON statistics for a specific Ethernet port or all Ethernet ports in the CSS. The CSS Enterprise ap64Stats MIB defines these statistics. You can also display the RFC1757 32-bit statistics by adding the **-32** suffix to the **show rmon** command.

- To display the RMON statistics for all ports in the CSS, enter:

```
# show rmon
```

To display the RFC1757 32-bit statistics, enter **show rmon-32**.

- To display the RMON statistics for a specified port in the CSS, enter:

```
# show rmon port_name
```

The *port_name* variable is the name of the physical port (for example, ethernet-4). Enter it as a case-sensitive unquoted text string.

To display the RFC1757 32-bit statistics, enter **show rmon-32** *port_name*.

To see a list of ports, enter:

```
# show rmon ?
```

For example, to display the extended RMON statistics for the Ethernet-4 port in the CSS, enter:

```
# show rmon ethernet-4
```

Table 10-4 lists and describes the fields in the **show rmon** output.

Table 10-4 Field Descriptions for the show rmon Command

Field	Description
Bytes	The total number of received bytes.
Packets	The total number of received packets (including bad packets, broadcast packets, and multicast packets).
Broadcast Packets	The total number of good received packets that were directed to the broadcast address. Note that this does not include multicast packets.
Multicast Packets	The total number of good received packets that were directed to a multicast address. This number does not include packets directed to the broadcast address.
CRC Alignment Errors	The total number of packets received that had a length (excluding framing bits, but including FCS octets) between 64 and 1518 octets, inclusive, but had either an FCS Error, a bad Frame Check Sequence (FCS) with an integral number of octets, or an Alignment Error, a bad FCS with a non-integral number of octets.
Oversize Packets	The total number of received packets that were longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed.
Undersize Packets	The total number of received packets that were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed.

Table 10-4 Field Descriptions for the show rmon Command (continued)

Field (continued)	Description
Fragments	<p>The total number of received packets that were less than 64 octets in length (excluding framing bits but including FCS octets) and had either an FCS Error, a bad Frame Check Sequence (FCS) with an integral number of octets, or an Alignment Error, a bad FCS with a non-integral number of octets.</p> <p>It is normal for fragment statistics to increment because the CSS counts both runts (which are normal occurrences due to collisions) and noise hits.</p>
Drop Events	The total number of events in which packets were dropped by the probe due to lack of resources. This number is not necessarily the number of packets dropped; it is the number of times this condition has been detected.
Slobbers	An internal counter. This field will always be zero.
Jabbers	<p>The total number of packets received that were longer than 1518 octets (excluding framing bits, but including FCS octets), and had either an FCS Error, a bad Frame Check Sequence (FCS) with an integral number of octets, or Alignment Error, a bad FCS with a non-integral number of octets.</p> <p>This definition of jabber is different than the definition in IEEE-802.3 section 8.2.1.5, 10BASE5, and section 10.3.1.4, 10BASE2. These documents define jabber as the condition where any packet exceeds 20 ms. The allowed range to detect jabber is between 20 ms and 150 ms.</p>

Table 10-4 Field Descriptions for the *show rmon Command* (continued)

Field (continued)	Description
Collisions	<p>The best estimate of the total number of collisions on this Ethernet segment.</p> <p>The returned value depends on the location of the RMON probe. Section 8.2.1.3, 10BASE-5, and section 10.3.1.3, 10BASE-2, of IEEE standard 802.3 states that a station must detect a collision, in the receive mode, if three or more stations are transmitting simultaneously. A repeater port must detect a collision when two or more stations are transmitting simultaneously. Thus, a probe placed on a repeater port might record more collisions than would a probe connected to a station on the same segment.</p> <p>Probe location plays a much smaller role when considering 10BASE-T. IEEE standard 802.3 14.2.1.4, 10BASE-T defines a collision as the simultaneous presence of signals on the DO and RD circuits (transmitting and receiving at the same time). A 10BASE-T station can only detect collisions when it is transmitting. Thus, probes placed on a station and a repeater should report the same number of collisions.</p> <p>Ideally, an RMON probe inside a repeater should report collisions between the repeater and one or more other hosts (transmit collisions as defined by IEEE 802.3k), plus receiver collisions observed on any coax segments to which the repeater is connected.</p>

Table 10-4 Field Descriptions for the `show rmon` Command (continued)

Field (continued)	Description
Packets (0-64)	The total number of packets (including bad packets) received that were between the following octets in length inclusive (excluding framing bits but including FCS octets):
Packets (65-127)	
Packets (128-255)	
Packets (256-511)	
Packets (512-1023)	
Packets (1024-1518)	<ul style="list-style-type: none"> • 0 to 64 • 65 to 127 • 128 to 255 • 256 to 511 • 512 to 1023 • 1024 to 1518

Clearing RMON Statistics

To reset the RMON statistics on a CSS Ethernet port to zero, use the **clear statistics *port_name*** command. The *port_name* variable is the name of the physical port (for example, ethernet-4). Enter it as a case-sensitive unquoted text string.

For example, to clear the statistics for Ethernet port 1, enter:

```
# clear statistics Ethernet-1
```

To see a list of ports, enter:

```
# clear statistics ?
```



Note

When you reset RMON statistics on a CSS Ethernet port to zero, the Ethernet errors and MIB-II statistics for the port are also reset to zero.

Viewing History

You can display the default and configured RMON history information for a specific Ethernet port or all Ethernet ports in the CSS. For information on configuring an RMON history, refer to “Configuring an RMON History” earlier in this chapter.

By default, the CSS maintains two tables of history statistics for each port. One table contains the last 50 samples at 30-second intervals. The other table contains 50 samples at 30-minute intervals. You cannot modify the configuration for these histories.

- To view the RMON history for all ports in the CSS, enter:

```
# show rmon-history
```

- To display the RMON history for a specified port, enter:

```
# show rmon-history port_name
```

To see a list of ports in the CSS, enter:

```
# show rmon-history ?
```

- To display the RMON history for a specified port and history index, enter:

```
# show rmon-history port_name history_index
```

For example, to view the history 5 for the Ethernet-4 port, enter:

```
# show rmon-history ethernet-4 5
```

To see a list of history indexes associated with a specified port, enter:

```
# show rmon-history port_name ?
```

For example, to see a list of histories for the Ethernet-4 port, enter:

```
# show rmon-history ethernet-4 ?
```

Table 10-5 lists and describes the fields in the **show rmon-history** output.

Table 10-5 Field Descriptions for the show rmon-history Command

Field	Description
Owner	The entity that configured the entry and is using the resources assigned to it.
Start Time	The time when the bucket sampling started.
Interval	The time interval in seconds when RMON takes a bucket sample.
Buckets	The number of discrete sampling intervals over which data is to be saved for the history.
Time	The time that the sample was taken.
Sample	The number of the sample.
Octets	<p>The total number of octets of data (including those in bad packets) received on the network, excluding framing bits but including FCS octets.</p> <p>You can use this object as a reasonable estimate of Ethernet utilization. If greater precision is desired, sample the Ethernet statistic packet and octet objects before and after a common interval. The differences in the sampled values are packets (Pkts) and Octets, respectively, and the number of seconds in the Interval. These values are used to calculate the utilization of a 10 MB Ethernet port as follows:</p> $\text{Utilization} = \frac{\text{Pkts} * (9.6 + 6.4) + (\text{Octets} * .8)}{\text{Interval} * 10,000}$ <p>The result of this equation is the utilization value, which is the utilization percentage of the Ethernet segment on a scale of 0 to 100 percent.</p>
Packets	The total number of received packets (including bad packets, broadcast packets, and multicast packets).
Errors	The total number of errors that RMON received for this port.
Util%	The bandwidth utilization percentage of the Ethernet segment on a scale of 0 to 100 percent.

Viewing Events in a Log File

The CSS can send notifications of RMON alarm events to a traplog file or a configured log location, such as a log file on the CSS disk, a CSS session, host syslog daemon, or an email address. The notification itself displays when the event occurred, the event number, and its configured description in parenthesis. For example:

```
FEB 15 15:41:22 EVENT#4 FIRED: (Service Toys exceeded 30,000
connections).
```

For information on configuring an RMON event, refer to “Configuring an RMON Event” earlier in this chapter. For information on configuring an RMON alarm, refer to “Configuring an RMON Alarm” earlier in this chapter.

Viewing a Traplog File

A trap log file is an ASCII file in the log directory containing generic and enterprise SNMP traps. No configuration is necessary. When an RMON alarm event occurs, a notification of its occurrence is saved in the trap log file on the CSS automatically.

**Note**

The traps sent to the traplog file are the same traps sent to an SNMP network management station. For information on configuring SNMP refer to Chapter 9, Configuring Simple Network Management Protocol (SNMP).

To display all SNMP traps that have occurred on the CSS, enter:

```
# show log traplog
```

**Note**

Even though traps are disabled, the CSS still produces a log message for any event that would normally generate a trap.

Viewing a CSS Disk Log File

Before the CSS can send an event to a log location, you must:

- Configure the location by using the **logging disk**, **host**, **line**, or **sendmail** command.
- Enable logging for the network management subsystem. To do so, enter:

```
(config)# logging subsystem netman level info-6
```

To view the events in a log file on the CSS disk, use the **show log log_filename** command. For example, to view a log file named log1, enter:

```
# show log log1
```

RMON Configuration in a Startup-Config File

The following example shows an RMON configuration in a startup-config file.

```

!***** RMON EVENT*****
rmon-event 1
  active
  description "Service connections exceeded 100"
  owner "Boston Tech Lab"
  community moonbase_alpha
  type log-and-trap

rmon-event 2
  active
  description "Service connections are below 90"
  owner "Boston Tech Lab"
  community moonbase_alpha
  type log-and-trap

!***** RMON ALARM *****
rmon-alarm 1
  active
  owner "Boston Tech Lab"
  sample-variable apSvcConnections.
  sample-type absolute
  startup-type rising-and-falling
  rising-threshold 100
  rising-event 1
  falling-threshold 90
  falling-event 1
  sample-interval 30

!***** RMON HISTORY *****
rmon-history 5
  active
  owner Boston Tech Lab
  data-source ifIndex.3
  interval 60
  requested-buckets 25

```