



# CHAPTER 7

## RADIUS Accounting

---

**Revised: March 20, 2009, OL-17222-03**

This chapter describes RADIUS Accounting in Cisco Access Registrar (CAR) as defined in Internet RFC 2866. This chapter contains the following sections:

- [Understanding RADIUS Accounting, page 7-1](#)
- [Setting Up Accounting, page 7-1](#)
- [Oracle Accounting, page 7-5](#)
- [MySQL Support, page 7-9](#)
- [Proxying Accounting Records, page 7-11](#)
- [Accounting Log Examples, page 7-13](#)
- [Sample Error Messages, page 7-14](#)

## Understanding RADIUS Accounting

RADIUS accounting is the process of collecting and storing the information contained in Accounting-Start and Accounting-Stop messages. Internet RFC 2866 describes the protocol for sending accounting information between a Network Access Server (NAS) and a RADIUS server (or shared accounting server).



**Note**

---

CAR uses UDP port number 1646 as its default port for RADIUS accounting messages. RFC 2866 defines UDP port number 1813 as the accounting port number.

---

When a NAS that uses accounting begins a session, it sends an Accounting-Start packet describing the type of service and the user being connected to the CAR server. When the session ends, the NAS sends the RADIUS server an Accounting Stop packet describing the type of service that was delivered. The Accounting Stop packet might also contain statistics such as elapsed time, input and output octets, or input and output packets.

## Setting Up Accounting

To configure CAR to perform accounting, you must do the following:

1. Create a service

2. Set the service type to file
3. Set the DefaultAccountingService field in **/Radius** to the name of the service you created

After you **save** and **reload** the CAR server configuration, the CAR server writes accounting messages to the **accounting.log** file in the **/opt/CSCOAr/logs** directory. The CAR server stores information in the **accounting.log** file until a rollover event occurs. A rollover event is caused by the **accounting.log** file exceeding a pre-set size, a period of time transpiring, or on a scheduled date.

When the rollover event occurs, the data in **accounting.log** is stored in a file named by the prefix *accounting*, a date stamp (*yyyymmdd*), and the number of rollovers for that day. For example, **accounting-20081107-14** would be the 14th rollover on November 07, 2008.

The following shows the properties for a service called CiscoAccounting:

```
[ //localhost/Radius/Services/CiscoAccounting ]
  Name = CiscoAccounting
  Description =
  Type = file
  IncomingScript~ =
  OutgoingScript~ =
  OutagePolicy~ = RejectAll
  OutageScript~ =
  FilenamePrefix = accounting
  MaxFileSize = "10 Megabytes"
  MaxFileAge = "1 Day"
  RolloverSchedule =
  UseLocalTimeZone = FALSE
```

## Accounting Log File Rollover

The CAR accounting functionality provides flexibility in managing the accounting log. You can configure the CAR server to rollover the accounting log using any combination of the following CAR accounting service properties:

- **MaxFileSize**— Indicates the maximum size of the accounting log file in KB, MB, or GB
- **MaxFileAge**— Indicates the maximum age of the log file in minutes, hours, days, or weeks
- **RolloverSchedule**— Indicates the exact time including the day of the month or day of the week, hour and minute to roll over the accounting log file

You can configure an accounting service using any combination of **MaxFileSize**, **MaxFileAge**, and **RolloverSchedule**. For example, you might configure **RolloverSchedule** and **MaxFileAge** at the same time. This would be useful if you wanted to have an age-based rollover, but also synchronize to an absolute clock at specified times. The following would set a rollover every twelve hours at 11:59 and 23:59.

```
set MaxFileAge "12 H"
```

```
set RolloverSchedule "59 11,23 * * *"
```

You might also consider scheduling **MaxFileAge** to be six minutes and set **RolloverSchedule** to the top of the hour. The following would create ten six-minute long files starting anew every hour.

```
set MaxFileAge "6 Minutes"
```

```
set RolloverSchedule "0 * * * *"
```

Although you specify an exact time with the RolloverSchedule property, the CAR server only checks the rollover schedule when an accounting event occurs. If your CAR server receives a steady flow of packets (at least one per minute), the times you specify are accurate. However, if the CAR server does not receive any packets for a period of time, no rollovers will occur until the next packet is received. The same is true for MaxFileAge and MaxFileSize.

Based on the maximum file size and the age specified, CAR closes the accounting file, moves it to a new name, and reopens the file as a new file. The name given to this accounting file depends on its creation and modification dates.

For example, if the file was created and modified on the same date, the filename will be of the format *FileNamePrefix-<yyyymmdd>-<n>.log*, and the suffix will have year, month, day, and number. If the file was created on some day and modified on another, the filename will be of the format *FileNamePrefix-<yyyymmdd>-<yyyymmdd>-<n>.log*, and the suffix will have creation date, modification date, and number.

## FilenamePrefix

The FileNamePrefix property enables you to specify a path to the file system in which you store the log files. If you do not manage your log files regularly, they might use the system resources, which will affect the performance of the CAR server. Cisco recommends that you store the log files in a file system different from the file system where you installed the CAR software by specifying the path in the FilenamePrefix property. By doing so the CAR server continues to run, even if the accounting logs fill the file system. The following example specifies the **/usr/arlogs/accounting** as the FilenamePrefix:

```
set /Radius/Services/CiscoAccounting/FilenamePrefix /usr/arlogs/accounting
```

You can also set up a *cron job* to check the size of the log files and mail the administrator if the file system is full.

## MaxFileSize

Use MaxFileSize to indicate the maximum size of the **accounting.log** file in minutes, hours, days, or weeks. MaxFileAge measures the age of the **accounting.log** file from the time the previous file rollover occurred.

You can specify the following (case insensitive) file sizes:

- K, Kilobytes, Kilobytes
- M, Megabyte, Megabytes
- G, Gigabyte, Gigabytes

The following are examples of valid commands to set MaxFileSize:

```
set MaxFileSize "500 kilobytes"
```

The example above sets a MaxFileSize of 500 kilobytes

```
set maxfilesize "1 G"
```

The example above sets a MaxFileSize of one gigabyte

```
set maxfilesize "200 megabyte"
```

The example above sets a MaxFileSize of 200 megabytes

## MaxFileAge

Use MaxFileAge to indicate the maximum age of the log file in minutes, hours, days, or weeks. MaxFileAge measures the age of the **accounting.log** file from the time the previous file rollover occurred.

You can specify the following (case insensitive) periods of time:

- M, Minute, or Minutes preceded by a number from 0 to 59
- H, Hour, or Hours preceded by a number from 0 to 23
- D, Day, or Days preceded by a number from 1 to 31
- W, Week, or Weeks preceded by a number from 1 to 52

The following are examples of valid commands to set MaxFileAge:

```
set MaxFileAge "6 Minutes"
```

The example above sets a MaxFileAge of 6 minutes.

```
set maxfileage "2 d"
```

The example above sets a MaxFileAge of two days.

```
set maxfileage "1 H"
```

The example above sets a MaxFileAge of one hour.

## RolloverSchedule

You set RolloverSchedule using the following crontab-style time format:

```
minute hour "day of month" "month of year" "day of week"
```

Where:

Minute is a value from 0-59

Hour is a value from 0-23

Day (of the month) is a value from 1-31

Month is a value from 1-12

Day (of the week) is a value from 0-6, where 0 is Sunday

## UseLocalTimeZone

When set to TRUE, the CAR server stores the accounting records in the log using the local system time. When set to FALSE (the default), CAR stores the accounting records in the log using Greenwich Mean Time (GMT).

# Oracle Accounting

Previous releases of CAR supported accessing user data from an Oracle database using Open Database Connectivity (ODBC), but this feature was limited to performing authentication and authorization (AA). You could only write the accounting records to local file or proxy to another RADIUS server. CAR supports writing accounting records into Oracle database enabling integration between billing systems and Oracle.

CAR adds a new type of service and remote server called *odbc-accounting* that enables inserting accounting records into Oracle. You can write accounting records into Oracle by referring this service in **/Radius/DefaultAccountingService** or in the Accounting-Service environment variable.

There is no specified schema structure to use the Oracle accounting feature. You can use your own table design and configure insert statements using standard SQL in the CAR configuration. The CAR server executes the insert statements to write the accounting record into Oracle. This feature is similar to the existing ODBC feature which performs authentication and authorization.

To improve latency for writing accounting records into database, packet buffering can be used. This option is enabled using the *BufferAccountingPackets* property under the *odbc-accounting* remote server definition.



## Note

Oracle 8i is no longer supported in AR4.2. However, CAR 4.2 supports Oracle 10g client and 11g server.

## Configuring Oracle Accounting

To use the Oracle accounting feature, you must configure a service of type *odbc-accounting* under **/Radius/Services**. You must also configure at least one remote servers of type *odbc-accounting* under **/Radius/RemoteServers**.

### ODBC-Accounting Service

The following is an example of an ODBC-Accounting service:

```
[ //localhost/Radius/Services/oracle_accounting ]
  Name = oracle_accounting
  Description =
  Type = odbc-accounting
  IncomingScript~ =
  OutgoingScript~ =
  OutagePolicy~ = RejectAll
  OutageScript~ =
  MultipleServersPolicy = Failover
  RemoteServers/
    1. accounting_server
```

### ODBC RemoteServers

Create a remote server under **/Radius/RemoteServers**, and set its protocol to *odbc-accounting*. The following is an example of an ODBC-Accounting RemoteServer's configuration:

```
[ //localhost/Radius/RemoteServers/accounting_server ]
  Name = accounting_server
  Description =
```

```

Protocol = odbc-accounting
ReactivateTimerInterval = 300000
Timeout = 15
DataSourceConnections = 8
ODBCDataSource =
KeepAliveTimerInterval = 0
BufferAccountingPackets = TRUE
MaximumBufferFileSize = "10 Megabytes"
NumberOfRetriesForBufferedPacket = 3
BackingStoreEnvironmentVariables =
UseLocalTimeZone = FALSE
AttributeList =
Delimiter =
SQLDefinition/
    
```

Table 7-1 describes the ODBC RemoteServer properties.

**Table 7-1 ODBC RemoteServer Properties**

Property	Description
Name	Name of the remote server; this property is mandatory, and there is no default
Description	Optional description of server
Protocol	Must be set to odbc-accounting
ReactivateTimerInterval	Mandatory time interval (in milliseconds) to activate an inactive server; defaults to 300000 ms.
Timeout	Mandatory time interval (in seconds) to wait for SQL operation to complete; defaults to 15 seconds
DataSourceConnections	Mandatory number of connections to be established; defaults to 8
ODBCDataSource	Name of the ODBCDataSource to use and must refer to one entry in the list of ODBC datasources configured under <b>/Radius/Advanced/ODBCDataSources</b> . Mandatory; no default
KeepAliveTimerInterval	Mandatory time interval to send a keepalive to keep the idle connection active; defaults to zero (0) meaning the option is disabled
BufferAccountingPackets	Mandatory, TRUE or FALSE, determines whether to buffer the accounting packets to local file, defaults to TRUE which means that packet buffering is enabled)
MaximumBufferFileSize	Mandatory if BufferAccountingPackets is set to TRUE, determines the maximum buffer file size, defaults to 10 Megabyte)
NumberOfRetriesForBufferedPacket	Mandatory if BufferAccountingPackets is set to TRUE. A number greater than zero determines the number of attempts to be made to insert the buffered packet into Oracle. Defaults to 3.
BackingStoreEnvironmentVariables	Optional; when BufferAccountingPackets is set to TRUE, contains a comma-separated list of environment variable names to be stored into a local file along with buffered packet. No default. BackingStoreEnvironmentVariables can also be specified in scripts using the BackingStoreEnvironmentVariables environment variable.
UseLocalTimeZone	Set to TRUE or FALSE, determines the timezone of accounting records' TimeStamp (defaults to FALSE).
AttributeList	List of comma-separated attribute names.

**Table 7-1 ODBC RemoteServer Properties (continued)**

Property	Description
Delimiter	Character used to separate the values of the attributes given in AttributeList property.
SQLDefinition	List of insert statements to be executed to insert the accounting record.

It is mandatory to set MaximumBufferSize property if BufferAccountingPackets property is set to TRUE. MaximumBufferSize can be specified in Kilobytes, Megabytes and Gigabytes. All values "512 KB", "512 kilobytes", "512 k", "512 K" are valid for specifying 512 kilobytes.

If buffering is enabled, incoming packets will be accepted and logged to local file until the configured buffer file size is reached even if the database is off-line. Attempts to insert them into Oracle will be made when database becomes available. This remote server will be marked as down only when the buffer gets full. So, having two odbc-accounting remote servers in the service, first one with buffering enabled and multiple server policy of FailOver will make the other remote servers to receive packets only when the first remote server's buffer gets full.

AttributeList is to specify the list of attribute names separated with comma. When this 'AttributeList' is given in the MarkerList, these attributes' values will be appended together with delimiter specified in 'Delimiter' property and will be supplied as input to that marker.

Attributes from the CAR environment and request dictionaries can be specified in the MarkerList. Request dictionary will be looked up first for the attributes. Other than the standard attributes in the CAR dictionaries, two new marker variables are supported inside the marker list. They are:

**TimeStamp** —Used to insert the timestamp into Oracle from CAR. Specifying this will supply the timestamp of that accounting record as a value to the insert statement. Time zone of this timestamp will be local if UseLocalTimeZone property is set to TRUE, otherwise GMT. This functionality could also be achieved by employing a trigger on the accounting table in the database. However, using this marker variable is recommended because the use of triggers negatively affects performance.

The format of the timestamp marker variable supplied by CAR is *YYYYMMDDHH24MMSS*. For example, a timestamp of 20081107211050 represents 21:10:50, November 07, 2008.

**RawAcctRecord**—Used to insert the entire accounting record into the database as a single text field. Contents of this will be whatever is sent by the NAS in the accounting packet and the format is *name=value* pairs delimited with the string specified in Delimiter property. If the delimiter property is not set, the default delimiter is a new line character. RawAcctRecord can be used with the other marker variables.

If multi-valued attributes are specified in the marker list, the multiple values are concatenated together with delimiters, and the resulting value will be passed to the insert statement. This delimiter can be specified using the ODBCEnvironmentMultiValueDelimiter property under **/Radius/Advanced**.

## Configuration Examples

This section provides common Oracle accounting configuration examples most likely to be used.

### Inserting Selected Attributes into Separate Columns

Use the following SQL and MarkerList properties statement to insert selected attributes into separate Oracle columns. The Oracle table definition will have separate columns for each attribute.

```
SQL: "insert into ar_acct (username,nasinfo,packet_type,timestamp) values (?, ?, ?, ?)"
```

```
MarkerList: "UserName/SQL_CHAR NAS-Identifier/SQL_CHAR Acct-Status-Type/SQL_CHAR
TimeStamp/SQL_TIMESTAMP"
```

In this example, all the column data types are CHAR/VARCHAR except the timestamp which is DATE. If packet buffering option is disabled, instead of TimeStamp marker, you can also use Oracle's **sysdate** as a value for the timestamp column. The insert statement will look like the following:

```
"insert into ar_acct (username,nasinfo,packet_type,timestamp) values (?,?=?,sysdate)"
```

### Inserting Complete Accounting Packets into One Column

Use SQL and MarkerList properties in the SQLStatement like the following to insert the complete accounting packet into one Oracle column.

```
SQL: "insert into ar_acct (timestamp,raw_packet) values (?,?)"
MarkerList: "TimeStamp/SQL_TIMESTAMP RawAcctRecord/SQL_VARCHAR"
```

### Inserting Selected Attributes into One Column

To insert selected attribute values into one Oracle column delimited by a comma (,), you must configure the AttributeList and Delimiter properties of the odbc-accounting RemoteServer object like the following:

```
AttributeList = "NAS-Identifier,NAS-Port,Acct-Status-Type,Acct-Session-Id"
Delimiter = ,
```

The SQL and MarkerList properties in the SQLStatement will look like the following:

```
SQL: "insert into ar_acct (username,timestamp,attributes) values (?,?=?,)"
MarkerList: "UserName/SQL_CHAR TimeStamp/SQL_TIMESTAMP AttributeList/SQL_VARCHAR"
```

## Packet Buffering

You can optionally use packet buffering to improve latency when writing accounting records into database. To enable packet buffering, set the BufferAccountingPackets property in the odbc-accounting remote server to TRUE.

### When Using Packet Buffering

When BufferAccountingPackets is set to TRUE, the CAR 4.2 server's Accounting-Response is returned as soon as the accounting record is successfully written to the local file. To accomplish the queuing of accounting records to a local file, a variant of the existing session backing store is used.

Buffered packets will be inserted into Oracle by a set of background worker threads. The CAR server tries to insert the buffered packet into Oracle for the number of retries configured in the NumberOfRetriesForBufferedPacket property (remote odbc accounting server definition). After the configured number of retries, the buffered packets are discarded from the local file.

Incoming packets will be buffered to local file until the configured MaximumBufferFileSize is reached. After this limit is reached, no more packets will be addressed. When the database is off-line, this remote server will continue to take incoming packets until MaximumBufferFileSize reaches. CAR tries to insert these buffered packets when database becomes available.

When using packet buffering, the CAR server can process more incoming packets and can reduce the bottleneck that could occur if the number of simultaneous incoming packets is large and the number of connections to the database is less.

## With Packet Buffering Disabled

When `BufferAccountingPackets` is set to `FALSE`, `Accounting-Response` is returned after writing the accounting record into Oracle. Oracle write timing is immediate.

Incoming packets are acknowledged by the remote server only after completing the write into Oracle.

When the database is off-line, no incoming packets are addressed. A slow database server impacts the packet processing rate.

## MySQL Support

CAR 4.2 provides support for MySQL to query user records from a MySQL database using `odbc` interface and enables you to write accounting records into MySQL database using `odbc-accounting`. CAR 4.2 has been tested with MySQL 4.0.18 and MyODBC 3.51.06 (reentrant).

For the CAR server to use MySQL, you must create and configure an `ODBCDataSource` object of type `myodbc` and a `RemoteServer` object set to protocol `odbc`.

## Configuring MySQL

To configure the CAR server to query records from a MySQL database, complete the following configuration:

- 
- Step 1** Log in to the CAR server and launch `aregcmd`.  
Log in as a user with administrative rights such as user `admin`.
- Step 2** Change directory to the `/Radius/Advanced/ODBCDataSources` and add a new `ODBCDataSource`.
- ```
cd /Radius/Advanced/ODBCDataSources
add mysql
```
- Step 3** Set the new `ODBCDataSource` type to `myodbc`.
- ```
cd mysql
set type myodbc
```
- Step 4** Set the `Driver` property to the path of the MyODBC library.
- Step 5** Set the `UserID` property to a valid username for the MyODBC database and provide a valid password for this user.
- Step 6** Provide a `DataBase` name and the name of the CAR `RemoteServer` object to associate with the `ODBCDataSource`.
- Step 7** Change directory to `/Radius/RemoteServers` and add a `RemoteServer` object to associate with the new `ODBCDataSource`.

```
cd /Radius/RemoteServers
```

```
add mysql
```

**Step 8** Change directory to the new RemoteServer and set its protocol to odbc-accounting.

```
cd mysql
```

```
set protocol odbc-accounting
```

**Step 9** Set the ODBCDataSource property to the name of the ODBCDataSource to associate with this RemoteServer object.

```
set ODBCDataSource mysql
```

---

## Example Configuration

The following shows an example configuration for a MySQL ODBC data source.

```
[ //localhost/Radius/Advanced/ODBCDataSources/mysql ]
Name = mysql
Type = myodbc
Driver = /tmp/libmyodbc3_r.so
UserID = mysql
Password = <encrypted>
DataBase = test
Server = mysql-a
Port = 3306
```

The following shows an example configuration for a RemoteServer

```
Name = odbc-accounting
Description =
Protocol = odbc-accounting
ReactivateTimerInterval = 300000
Timeout = 15
DataSourceConnections = 8
ODBCDataSource =
KeepAliveTimerInterval = 0
BufferAccountingPackets = TRUE
MaximumBufferFileSize = "10 Megabytes"
NumberOfRetriesForBufferedPacket = 3
BackingStoreEnvironmentVariables =
UseLocalTimeZone = FALSE
AttributeList =
Delimiter =
SQLDefinition/
ODBCToRadiusMappings/
ODBCToEnvironmentMappings/
ODBCToCheckItemMappings/
```

# Proxying Accounting Records

You can configure CAR to store accounting records locally and to proxy the accounting records to a remote RADIUS server thereby maintaining multiple accounting logs.

## Configuring the Local Cisco Access Registrar Server

This type of setup requires you to configure the following on the local CAR server:

- A local accounting service of type file
- A remote accounting service of type radius
- An accounting service of type group
- A RemoteServer object

## Configuring the Local Accounting Service

The following example shows the configuration required for a local accounting service. This service must be of type file.

```
[//localhost/Radius/Services/accserv1/ ]
  Name = accserv1
  Description =
  Type = file
  IncomingScript~ =
  OutgoingScript~ =
  OutagePolicy~ = RejectAll
  OutageScript~ =
  FilenamePrefix = accounting
  MaxFileSize = "10 Megabytes"
  MaxFileAge = "1 Day"
  RolloverSchedule =
  UseLocalTimeZone = FALSE
```

## Configuring the Remote Accounting Service

The following example shows the configuration required for a remote accounting service. This service must be of type *radius*, and the name of the remote server must be listed under the RemoteServers subdirectory.

```
[//localhost/Radius/Services/accserv2/
  Name = accserv2
  Description =
  Type = radius
  IncomingScript~ =
  OutgoingScript~ =
  OutagePolicy~ = RejectAll
  OutageScript~ =
  MultipleServersPolicy = Failover
  RemoteServers/
    1. RemoteRADIUS
```

## Configuring the Group Accounting Service

The following example shows the configuration required for a grouping accounting service. This service must be of type `group` and the local and remote accounting services, `accserv1` and `accserv2` in the previous examples, should be added under the `GroupServices` subdirectory.

The `CiscoAccounting` service groups these two services. The type property should be set to `group`. The services `accserv1` and `accserv2` should be added under `GroupServices` subdirectory of `CiscoAccounting` service.

```
[//localhost/Radius/Services/GroupAccounting/
  Name = GroupAccounting
  Description =
  Type = group
  IncomingScript~ =
  OutgoingScript~ =
  RolloverSchedule =
  ResultRule = AND
  GroupServices/
    1. accserv1
    2. accserv2
```

Refer to [Service Grouping Feature, page 16-13](#), for more information about the CAR Service Grouping feature.

## Configuring the RemoteServer Object

The following example shows the configuration required for the `RemoteServer` object in the local CAR server.

```
[ //localhost/Radius/RemoteServers ]
  Entries 1 to 1 from 1 total entries
  Current filter: <all>

  RemoteRADIUS/
    Name = RemoteRADIUS
    Description =
    Protocol = radius
    IPAddress = aa.bb.cc.dd
    Port = 1645
    ReactivateTimerInterval = 300000
    SharedSecret = secret
    Vendor =
    IncomingScript~ =
    OutgoingScript~ =
    MaxTries = 3
    InitialTimeout = 2000
    AccountingPort = 1646
    ACKAccounting = TRUE
```

If the `ACKAccounting` property is set to `FALSE`, CAR disregards the accounting acknowledgement and continues with the packet processing rather than waiting for the accounting acknowledgement from the Remote server.

The group service, `CiscoAccounting` in this example, should be defined as the default accounting service for any accounting packets received by the local CAR server, as in the following:

```
set /Radius/DefaultAccountingService CiscoAccounting
```

# Accounting Log Examples

This section provides examples of accounting log information recorded in an accounting log file.

## Accounting-Start Packet

The Accounting-Start packet describes the type of service and the user attempting to login.

```
Mon, 05 May 2008 05:23:02
  User-Name = bob%ppp
  NAS-Port = 71
  NAS-Identifier = localhost
  Acct-Status-Type = Start
  Acct-Session-Id = S209524
```

## Accounting Stop Packet

When the session ends, the NAS sends an Accounting Stop packet that describe the type of service that was delivered. The Accounting Stop packet might also contain statistics such as elapsed time, input and output octets, or input and output packets.

```
Thu, 15 May 2008 04:45:30
  User-Name = bob%PPP
  NAS-Port = 181
  NAS-Identifier = localhost
  Acct-Status-Type = Stop
  Acct-Session-Id = S209524
```

## Trace of Successful Accounting

The following is a trace example of a a successful accounting sequence.

```
11/23/200811/23/2008 21:27:58: P6699: Packet received from 10.1.9.204
11/23/2008 21:27:58: P6699: Trace of Accounting-Request packet
11/23/2008 21:27:58: P6699:   identifier = 237
11/23/2008 21:27:58: P6699:   length = 45
11/23/2008 21:27:58: P6699:   reqauth = ed:d6:a6:ae:57:09:b8:55:a8:d4:c4:0d:f7:be:06:2a
11/23/2008 21:27:58: P6699:   User-Name = bob
11/23/2008 21:27:58: P6699:   NAS-Identifier = localhost
11/23/2008 21:27:58: P6699:   Acct-Status-Type = Start
11/23/2008 21:27:58: P6699:   Acct-Session-Id = 1
11/23/2008 21:27:58: P6699: Using Client: cubone (10.1.9.204)
11/23/2008 21:27:58: P6699: Using NAS: localhost (127.0.0.1)
11/23/2008 21:27:58: P6699: Request is directly from a NAS: FALSE
11/23/2008 21:27:58: P6699: Running NAS localhost (127.0.0.1) IncomingScript: Pa seServiceHints
11/23/2008 21:27:58: P6699:   Rex: environ->get( "Request-Type" ) -> "Accounting-Request"
11/23/2008 21:27:58: P6699:   Rex: environ->get( "User-Name" ) -> ""
11/23/2008 21:27:58: P6699:   Rex: request->get( "User-Name", 0 ) -> "bob"
11/23/2008 21:27:58: P6699: Accounting with Service accserv1
11/23/2008 21:27:58: P6699: Trace of Accounting-Response packet
11/23/2008 21:27:58: P6699:   identifier = 237
11/23/2008 21:27:58: P6699:   length = 20
11/23/2008 21:27:58: P6699:   reqauth = a6:40:45:02:4c:8b:6f:00:4f:18:4a:b8:fe:28:9d:f4
11/23/2008 21:27:58: P6699: Sending response to 10.1.9.204
```

# Sample Error Messages

The following are sample accounting error messages:

**Error message logged in name\_radius\_1\_log file when the disk is full and AR is trying to record an accounting request.**

```
05/15/2008 2:52:29 name/radius/1 Error System 0 Failed to write records to the accounting report file '/usr/accounting.log' - accounting records lost
```

**Note**

---

An Accounting-Response packet is sent only if the accounting record is written to the file in the disk. If the disk is full, an Accounting-Response packet is not sent.

---

**Error message logged in name\_radius\_1\_log file when the path specified in the FilenamePrefix property is not valid.**

```
05/15/2008 4:11:12 name/radius/1 Error Configuration 0 Error in property /Radius/Services/CiscoAccounting/FilenamePrefix: Unable to write to the specified report file prefix (/tmp/AR/accounting)
```