NAME

radium.conf – radium resource file.

SYNOPSIS

radium.conf

DESCRIPTION

Radium will open this radium.conf if its installed as /etc/radium.conf. It will also search for this file as radium.conf in directories specified in $RADIUMPATH, or $RADIUMHOME, $RADIUMHOME/lib, or $HOME, $HOME/lib, and parse it to set common configuration options. All values in this file can be overriden by command line options, or other files of this format that can be read in using the -F option.

Variable Syntax

Variable assignments must be of the form:
  VARIABLE=
with no white space between the VARIABLE and the '=' sign. Quotes are optional for string arguments, but if you want to embed comments, then quotes are required.

RADIUM_DAEMON

Radium is capable of running as a daemon, doing all the right things that daemons do. When this configuration is used for the system daemon process, say for /etc/radium.conf, this variable should be set to "yes".

The default value is to not run as a daemon.

This example is to support the ./support/Startup/radium script which requires that this variable be set to "yes".

Commandline equivalent -d

RADIUM_DAEMON=no

RADIUM_MONITOR_ID

Radium Monitor Data is uniquely identifiable based on the source identifier that is included in each output record. This is to allow you to work with Argus Data from multiple monitors at the same time. The ID is 32 bits long, and supports a number of formats as legitimate values. Radium supports unsigned ints, IPv4 addresses and 4 bytes strings, as values.

The formats are discerned from the values provided. Double-quoted values are treated as strings, and are truncated to 4 characters. Non-quoted values are tested for whether they are hostnames, and if not, then they are tested wheter they are numbers.

The configuration allows for you to use host names, however, do have some understanding how ‘hostname‘ will be resolved by the nameserver before commiting to this strategy completely.

For convenience, argus supports the notion of "‘hostname‘" for assigning the probe's id. This is to support management of large deployments, so you can have one argus.conf file that works for a lot of probes.

For security, argus does not rely on system programs, like hostname.1. It implements the logic of hostname itself, so don't try to run arbitrary programs using this method, because it won't work.

Commandline equivalent -e

RADIUM_MONITOR_ID=‘hostname‘ // IPv4 address returned **RADIUM_MONITOR_ID**=10.2.45.3 // IPv4 address **RADIUM_MONITOR_ID**=2435 // Number **RADIUM_MONITOR_ID**="en0"

// String

RADIUM_ARGUS_SERVER

Radium can attach to any number of remote argus servers, and collect argus data in real time. The syntax for this variable is a hostname or a dot notation IP address, followed by an optional port value, separated by a ':'. If the port is not specified, the default value of 561 is used.

Commandline equivalent -S <host[:port]>

RADIUM_ARGUS_SERVER=localhost:561

RADIUM_CISCONETFLOW_PORT

Radium can read Cicso Netflow records directly from Cisco routers. Specifying this value will alert Radium to open a UDP based socket listening for data from this name or address.

Commandline equivalent -C

RADIUM_CISCONETFLOW_PORT=9996

RADIUM_USER_AUTH, RADIUM_AUTH_PASS

When argus is compiled with SASL support, ra* clients may be required to authenticate to the argus server before the argus will accept the connection. This variable will allow one to set the user and authorization id's, if needed. Although not recommended you can provide a password through the RADIUM_AUTH_PASS variable. The format for this variable is:

Commandline equivalent -U

RADIUM_USER_AUTH=user_id/authorization_id **RADIUM_AUTH_PASS=**the_password

RADIUM_ACCESS_PORT

Radium monitors can provide a real-time remote access port for collecting Radium data. This is a TCP based port service and the default port number is tcp/561, the "experimental monitor" service. This feature is disabled by default, and can be forced off by setting it to zero (0).

When you do want to enable this service, 561 is a good choice, as all ra* clients are configured to try this port by default.

Commandline equivalent -P

RADIUM_ACCESS_PORT=561

RADIUM_BIND_IP

When remote access is enabled (see above), you can specify that Radium should bind only to a specific IP address. This is useful, for example, in restricting access to the local host, or binding to a private interface while capturing from another. The default is to bind to any IP address.

Commandline equivalent -B

RADIUM_BIND_IP="127.0.0.1"

RADIUM_OUTPUT_FILE

Radium can write its output to one or a number of files, default limit is 5 concurrent files, each with their own independant filters.

The format is:

```
RADIUM_OUTPUT_FILE=/full/path/file/name
RADIUM_OUTPUT_FILE=/full/path/file/name "filter"
```

Most sites will have radium write to a file, for reliablity and performance. The example file name is used here as supporting programs, such as ./support/Archive/radiumarchive are configured to use this file.

Commandline equivalent -w

RADIUM_OUTPUT_FILE=/var/log/radium/radium.out

RADIUM_SET_PID

When Radium is configured to run as a daemon, with the -d option, Radium can store its pid in a file, to aid in managing the running daemon. However, creating a system pid file requires priviledges that may not be appropriate for all cases.

When configured to generate a pid file, if Radium cannot create the pid file, it will fail to run. This variable is available to override the default, in case this gets in your way.

The default value is to generate a pid.

No Commandline equivalent

RADIUM_SET_PID=yes

RADIUM_ADJUST_TIME

Radium can correct for time synchronization problems that may exist between data sources. If configured to do so, radium will adjust all the timestamps in records by the calculated drift between radium and its many data sources. Records whose timevalues have been 'corrected' are marked so that subsequent readers can differentiate between true primitive time and modified time.

Commandline equivalent -T

RADIUM_ADJUST_TIME=no

RADIUM_MAR_STATUS_INTERVAL

Radium will periodically report on a its own health, providing interface status, total packet and bytes counts, packet drop rates, and flow oriented statistics.

These records can be used as "keep alives" for periods when there is no network traffic to be monitored.

The default value is 300 seconds, but a value of 60 seconds is very common.

Commandline equivalent -M

RADIUM_MAR_STATUS_INTERVAL=60

RADIUM_DEBUG_LEVEL

If compiled to support this option, Radium is capable of generating a lot of debug information.

The default value is zero (0).

Commandline equivalent -D

RADIUM_DEBUG_LEVEL=0

RADIUM_FILTER_OPTIMIZER

Radium uses the packet filter capabilities of libpcap. If there is a need to not use the libpcap filter optimizer, you can turn it off here. The default is to leave it on.

Commandline equivalent -O

RADIUM_FILTER_OPTIMIZER=yes

RADIUM_FILTER

You can provide a filter expression here, if you like. It should be limited to 2K in length. The default is to not filter.

No Commandline equivalent

RADIUM_FILTER=""

RADIUM_CHROOT_DIR

Radium supports chroot(2) in order to control the file system that radium exists in and can access. Generally used when radium is running with privleges, this limits the negative impacts that radium could inflict on its host machine.

This option will cause the output file names to be relative to this directory, and so consider this when trying to find your output files.

Commandline equivalent -C

RADIUM_CHROOT_DIR=""

RADIUM_SETUSER_ID

Radium can be directed to change its user id using the setuid() system call. This is can used when radium is started as root, in order to access privleged resources, but then after the resources are opened, this directive will cause radium to change its user id value to a 'lesser' capable account. Recommended when radium is running as a daemon.

Commandline equivalent -u

RADIUM_SETUSER_ID="user"

RADIUM_SETGROUP_ID

Radium can be directed to change its group id using the setgid() system call. This is can used when radium is started as root, in order to access privleged resources, but then after the resources are opened, this directive can be used to change argu's group id value to a 'lesser' capable account. Recommended when radium is running as a daemon.

Commandline equivalent -g

RADIUM_SETGROUP_ID="group"

RADIUM_CLASSIFIER_FILE

Radium can be used to label records as they are distributed. This can be used to classify flow records, or simply to mark them for post processing purposes.

When provided with a ralabel.conf formatted file, radium will label all matching records.

Commandline equivalent none

RADIUM_CLASSIFIER_FILE=/usr/local/argus/ralabel.conf

RADIUM_CORRELATE

Radium has a correlation function, where flow data from multiple source's can be compared and 'correlateda.

This function is enabled with a single radium configuration keyword RADIUM_CORRELATE="yes". With this variable set, radium(). will buffer incoming data to generate delay, and will correlate data from multiple sources with an event window of about 3 seconds. Data that is matchable, which means that it has the same flow identifiers, or the same hints, will treated as if they were "observed" by multiple probes, and merged.

Commandline equivalent none

RADIUM_CORRELATE="no"

COPYRIGHT

SEE ALSO

radium(8)