NAME

nafload − NetSA Aggregated Flow RDBMS loader

SYNOPSIS

```
nafload     [--in INPUT_SPECIFIER] [--out OUTPUT_SPECIFIER]
            [--nextdir PROCESSED_INPUT_DIRECTORY]
            [--faildir FAILED_INPUT_DIRECTORY]
            [--poll POLLING_DELAY] [--lock]
            [--log LOG_SPECIFIER] [--loglevel LOG_LEVEL]
            [--verbose] [--version] [--daemon] [--foreground]
            [--sid SOURCE_ID] [--label LABEL] [--autolabel]
            [SQL_STATEMENT]
```

DESCRIPTION

nafscii takes NAF aggregated flow format files as input, and inserts their contents into a relational database. If NAF was built without AirDBC support, nafload simply prints a message to this effect and exits.

nafload, like all NAF tools, operates by default in **once** mode, though it can also be run as a **daemon**. In daemon mode, nafload will wait for new input to match its input specifier, and move processed input to the **−−nextdir** directory. This can be used to build ‘‘chains’’ of daemons for automated batch processing of flow data.

OPTIONS

Input Options

The input specifier determines where nafscii will read its input from. nafscii defaults to reading from standard input.

−−in *INPUT_SPECIFIER*

INPUT_SPECIFIER is an input specifier. This is a filename, a directory name, a file glob pattern (in which case it should be escaped or quoted to prevent the shell from expanding the glob pattern), or the string − to read from standard input.

Output Options

The output specifier determines the database to which nafload will connect to load NAF records. It is required. The other output options determine how the **:label** and **:srcid** parameters will be bound in the SQL expression

−−out *OUTPUT_SPECIFIER*

OUTPUT_SPECIFIER is an AirDBC URL specifying the database to load NAF records into. The AirDBC URL takes the following form:

*driver***://**[*username*[**:***password*]**@**]*hostname*[**:***port*]/*name*

where *driver* is one of:

oci Oracle 9i+ via OCI

postgresql
Postgresql 8+ via libpq4

and *name* is the database name to pass to the RDBMS.

−−sid *SOURCE_ID*

Force the source ID field on inserted records to the given 32−bit integer.

−−label *LABEL*

Force the **:label** in the SQL statement to the value in *LABEL*.

−−autolabel

If present, assume each output file was created by *nafalize*(1) and set the **:label** field of each executed SQL statement from the aggregation label present in the input filename.

Daemon Options

These options are used to run nafscii in daemon mode for batch processing of packet and flow files.

−−daemon
Run nafscii in daemon mode. Instead of processing its input then exiting, nafscii will continually look for new input matching its input specifier. This will cause nafscii to fork into the background and exit.

−−foreground
Instead of forking in **−−daemon** mode, stay in the foreground. Useful for debugging.

−−lock
Use lockfiles for concurrent file access protection. Highly recommended in **−−daemon** mode, especially if two NAF daemons are interacting through a given directory.

−−poll *POLLING_DELAY*
POLLING_DELAY is the polling delay in seconds; how long nafscii will wait for new input when none is available. The default is 60 seconds.

−−nextdir *PROCESSED_INPUT_DIRECTORY*
When reading from files, if this option is present, input files will be moved to *PROCESSED_INPUT_DIRECTORY* after they are successfully processed. The special string **delete** will cause successfully processed input to be removed instead. This option is required in daemon mode.

−−faildir *FAILED_INPUT_DIRECTORY*
When reading from files, if this option is present, input files will be moved to *FAILED_INPUT_DIRECTORY* if processing failed. The special string **delete** will cause failed input to be removed instead. This option is required in daemon mode.

Logging Options

These options are used to specify how log messages are routed. nafalize can log to standard error, regular files, or the UNIX syslog facility.

−−log *LOG_SPECIFIER*
Specifies destination for log messages. *LOG_SPECIFIER* can be a *syslog* (3) facility name, the special value **stderr** for standard error, or the *absolute* path to a file for file logging. Standard error logging is only available in **−−daemon** mode if **−−foreground** is present. The default log specifier is **stderr** if available, **user** otherwise.

−−loglevel *LOG_LEVEL*
Specify minimum level for logged messages. In increasing levels of verbosity, the supported log levels are **quiet**, **error**, **critical**, **warning**, **message**, **info**, and **debug**. The default logging level is **warning**.

−−verbose
Equivalent to **−−loglevel debug**.

−−version
If present, print version and copyright information to standard error and exit.

SQL STATEMENT SYNTAX

If present, the *SQL_STATEMENT* on the nafload command line should be a single AirDBC-syntax parameterized prepared statement. AirDBC uses Oracle-style parameters of the form **:***name*. The following parameters will be bound by nafload:

:stime
Bin time in UTC, ISO 8601 format

:etime
Bin time in UTC plus bin length, ISO 8601 format

:srcid
Source ID, unsigned integer

:sip Source IPv4 address, unsigned integer

:sipmask
Source IPv4 mask length, unsigned integer

:dip
Destination IPv4 address, unsigned integer

:dipmask
Destination IPv4 mask length, unsigned integer

:sp Source transport port, unsigned integer

:dp Destination transport port or ICMP type and code, unsigned integer

:proto
IP protocol, unsigned integer

:host
Unique source host count, unsigned integer

:rhost
Unique destination host count, unsigned integer

:port
Unique source port count, unsigned integer

:rport
Unique destination port count, unsigned integer

:flo Forward flow count, unsigned integer

:rflo
Reverse flow count, unsigned integer

:pkt
Forward packet count, unsigned big (64−bit) integer

:rpkt
Reverse packet count, unsigned big (64−bit) integer

:oct Forward octet count, unsigned big (64−bit) integer

:roct
Reverse octet count, unsigned big (64−bit) integer

:label
Record label, from **−−label** or **−−autolabel** command line option

If no *SQL_STATEMENT* is supplied, nafload will use a default suitable for the AirCERT v2 schema:

```
SELECT air_naf_insert(:stime, :etime, :srcid,
                      :sip, :sipmask, :dip, :dipmask, :sp, :dp, :proto,
                      :host, :rhost, :port, :rport,
                      :flo, :rflo, :pkt, :rpkt, :oct, :roct, :label)
```

nafload will execute the statement once for each record, so if your schema has multiple tables for each NAF record (as the AirCERT v2 schema does), it is suggested that you use a stored procedure to split the NAF record and synchronize internal identifiers properly. Indeed, this is precisely what the default statement does.

Note that if a column is not present in the NAF input data, that is, if it was not present in the *nafalize* (1) aggregation expression, it will be bound as NULL in the inserted row.

SIGNALS

nafload responds to **SIGINT** or **SIGTERM** by terminating input processing and exiting.

BUGS

Known issues are listed in the **README** file in the NAF tools source distribution. Note that NAF should be considered alpha-quality software; not every concievable input and aggregation is exhaustively tested at each release, and specific features may be completely untested. Please be mindful of this before deploying NAF in production environments. Bug reports and feature requests may be sent directly to the author, Brian Trammell, via email at <bht@cert.org>.

AUTHORS

Brian Trammell <bht@cert.org>, for the CERT Network Situational Awareness Group, http://www.cert.org/netsa.

SEE ALSO

nafalize (1), *nafilter* (1)