

NAME

proclog - interface program for the proclog package

SYNOPSIS

```
proclog -v
proclog -k killcmd
proclog -c [-ls] [messagesfile]
proclog -n [-ls] rotatedfile
proclog -m [-ls] [-o outfile] [messagesfile]
proclog -d [-ls] label
proclog -r [-ls] [-i infile] [-o output] [-b bday] [-e eday]
      [-z lines] [-p period] report-name
proclog -x [-ls] [-i file] [-o output] [-b bday] [-e eday]
      [-p period] -t type report-name
```

DESCRIPTION

The `proclog` program is the user interface program for using the functionality of the `proclog(7)` package.

The `proclog` package is the main administrative log processing engine on the firewall. It will transform the logging subsystem messages generated by the packages/services on the firewall into a name/value pair based intermediate-record, for example:

```
v=1 dat=20000226002715 sys=ns srv=txp usr=unknown \  
      src=192.168.20.30 dst=192.168.20.11 por=8080 sts=DENY
```

The first four name/value pairs are always available, they specify the intermediate file format version, the date when a specific service was started, the system on which the message was generated, and the service that generated the message. The other fields depend on the type of service.

At some later time, these intermediate-records are used for report generation and data export. Furthermore, these records can be archived to generate long term reports at some later point in time.

Preprocessing

The process of transforming messages into the intermediate-records is based on specific knowledge. The program has to know in which order the messages for, say, a HTTP browsing session will appear in the logfile, to be able to collect the appropriate information and pack it into a single, new record. Therefore `proclog` is built in a modular fashion. Firewall packages can add the additional modules needed to handle their messages properly.

Intermediate-records represent a complete session. This session may be related to more than one message in the messagesfile. Proclog collects the messages belonging to a certain session and uses these to produce the intermediate-record.

Generic modules are located in the `proclog mod` subdirectory and have extension `.pm`. Package specific modules are located in the `proclog` subdirectory of the package and also have extension `.pm`. Filenames must be globally unique. To disable a certain module, it should be renamed with different extension, e.g. `.pm-`.

The following modules are part of the standard `proclog` package:

<code>auth.pm</code>	Authsrv message processing
<code>ftp.pm</code>	FTP proxy message processing
<code>http.pm</code>	HTTP proxy message processing
<code>isdn.pm</code>	ISDN router message processing (currently only Ascend/Lucent)
<code>netacl.pm</code>	Network service message processing
<code>smap.pm</code>	Mail relay message processing
<code>smartfilter.pm</code>	Smartfilter/smartie message processing
<code>socks.pm</code>	SOCKS daemon message processing
<code>telnet.pm</code>	Telnet proxy message processing
<code>txp.pm</code>	TXP message processing

Proclog module programming is discussed in a separate document.

Post Processing

There are basically two methods for post processing the intermediate-records: report generation and data export.

Reports

Report generation produces ASCII records. The layout of these reports is predetermined and, most often, not customizable. Generic report definitions are found in the proclog `rpt` subdirectory and have extension `.rpt`. Package specific report definitions are found in the proclog subdirectory of the package and also have extension `.rpt`. Report definitions must be globally unique. The following report definitions are part of the standard proclog package:

<code>auth.rpt</code>	Authentication report
<code>ftp.rpt</code>	FTP proxy accounting report
<code>http.rpt</code>	HTTP proxy accounting report
<code>isdn.rpt</code>	ISDN accounting report
<code>netacl.rpt</code>	Netacl accounting report
<code>smap.rpt</code>	Mail relay accounting report
<code>smartfilter.rpt</code>	Smartfilter (url blocking) report
<code>socks.rpt</code>	Socks accounting report
<code>telnet.rpt</code>	Telnet proxy accounting report
<code>txp.rpt</code>	TXP accounting report
<code>user.rpt</code>	User/section accounting report

Exports

Data export produces output in some specific export format. Data is actually generated by “export-generators”, which use export definitions to determine what needs to be exported. The generators can be found in the proclog `gen` subdirectory and have extension `.gen`. The following export generators are part of the standard proclog package:

<code>clf.gen</code>	Common Logfile Format generator
<code>csv.gen</code>	Comma Separated Values generator
<code>dbf.gen</code>	DataBase Format generator
<code>welf.gen</code>	WebTrends logging format generator

The layout of the export files depends on the generator only and is customizable through the export definitions. Export definitions are found in the proclog `exp` subdirectory and have extension

`.exp`. Package specific export definitions are located in the `proclog` subdirectory of the package and also have extension `.exp`. Export definitions must be globally unique. The following export definitions are part of the standard `proclog` package:

<code>auth.exp</code>	Authentication export definition
<code>ftp.exp</code>	FTP proxy accounting export definition
<code>http.exp</code>	HTTP proxy accounting export definition
<code>isdn.exp</code>	ISDN accounting export definition
<code>netacl.exp</code>	Netacl accounting export definition
<code>smartfilter.exp</code>	Smartfilter accounting export definition
<code>smtp.exp</code>	SMTP plug accounting export definition
<code>socks.exp</code>	Socks accounting export definition
<code>telnet.exp</code>	Telnet proxy accounting export definition
<code>txp.exp</code>	TXP accounting export definition

The export definition file consists of lines which define the fields of the intermediate-records to be exported. All lines starting with a '#' sign are interpreted as comment and empty lines are skipped. The other lines have the following syntax:

SERVICE servicename

This line defines the service to use (i.e. the service field of an intermediate-record to look for). Exactly one `SERVICE` line must be present. The following example specifies the selection of `auth` (authentication) records.

```
SERVICE auth
```

DEFINE field-name, export-name, field-type, field-length

This line defines a field to be exported. At least one `DEFINE` line must be present. The order of the `DEFINES` specifies the order of the exported data (usually the column order). The following example specifies the selection of the "nam" and "siz" fields of an intermediate-record. "Username" and "Total" are the corresponding headers. These headers are however only used if this function is provided by the generator. The first field is a 30 bytes string and the second an 8 bytes number. The "nam" field will appear first in the data export.

```
DEFINE nam,Username,string,30
DEFINE siz,Total,num,8
```

NOTE: in this version of the export definition file, the name of the file **MUST** be the same as the name of the servicename on the `SERVICE` line. This restriction will disappear in some future version.

Options

The `proclog` program has a number of command line switches which can be used with most of the `proclog` modes:

- l This switch turns `proclog`'s internal logging on. `Proclog` will log what it is doing in the standard `proclog` logfile: `/system/proclog/log/proclog.log`. The main purpose of this switch is the gathering of more information for debugging in case `proclog` execution fails for some reason.
- s Depending on the mode, `proclog` prints warnings and errors on `STDERR` or to the logging subsystem. This switch is used to force warning and error messages to the logging subsystem. This can be handy whenever some "manual" mode application which sends its errors to `STDERR` (like the `report` mode) is used within some script started from `cron`.

Default behavior:

crontab mode	logging subsystem
newsyslog mode	logging subsystem
manual mode	STDERR
report mode	STDERR
export mode	STDERR
docset mode	logging subsystem
kill mode	logging subsystem

The program operates in a number of different modes, which are activated by program switches. One and only one mode should be present at any invocation. The rest of this section describes the different `proclog` modes and their specific options.

version mode

```
proclog -v
```

This mode makes `proclog` print its version number and copyright info.

crontab mode

```
proclog -c [-ls] [messagesfile]
```

Crontab mode is used for regular generation of intermediate records from the messages. Every run will start at the point where a previous proclog (crontab or newsyslog) run ended. State information is kept to accomplish this, because a certain session may not have finished when the proclog run ends.

The resulting intermediate-records for a specific day are written to the proclog data area, where the files have names in Julian date (YYYYDDD) format.

Crontab mode (and newsyslog mode) will set a lock, so a new instance of the program won't start processing until the current one has finished.

Crontab mode is typically started from cron every night or more frequently. The optional argument specifies which *messagesfile* to process (default: /var/log/messages).

This mode will write warnings and other messages to the logging subsystem.

newsyslog mode

```
proclog -n [-ls] rotatedfile
```

Newsyslog mode is used by the newsyslog program to let proclog process the latest rotated messages (since the last checkpoint). Processing is exactly the same as in crontab mode. The mandatory argument specifies the rotated, possibly gzipped, *messagesfile*.

This mode will write warnings and other messages to the logging subsystem.

manual mode

```
proclog -m [-ls] [-o outfile] [messagesfile]
```

Manual mode is mostly used for interactive usage. Processing is almost the same as in crontab mode. There are however a few significant differences: the intermediate-records are written to STDOUT and no state info or checkpoints are used or generated. This means that processing is independent from crontab/newsyslog mode processing and the complete *messagesfile* is read at every invocation.

Manual mode is typically used for testing and generation of intermediate records from archived messagesfiles. The optional argument specifies the *messagesfile* to use (default: /var/log/messages).

The -o *outfile* option specifies that the intermediate-records should be written to *outfile* instead of STDOUT.

This mode will write warnings and messages to STDERR.

document set (docset) mode

```
proclog -d [-ls] label
```

Docset mode is the mode which produces all the periodical documents. This mode is typically started from `cron` at regular intervals. The only mandatory argument is the docset *label* to use. This *label* specifies the configuration entries to process. E.g. if `proclog` is started in docset mode with label “daily”, then all configuration entries with *label* “daily” will be processed.

Configuration entries are found in the global `proclog` configuration file `config.proclog`. Users of the firewall can add their own entries to this file. Package specific configuration entries located in the `proclog` subdirectory of the packages, in files with extension `.set`. The format of the configuration lines in both files is identical.

The format of the docset configuration entries are:

```
REPORT          "docname" "label" "destination" ...
EXPORT "type" "docname" "label" "destination" ...
```

Example:

```
REPORT          http  daily  joe@acme.nl
EXPORT csv      auth   weekly jack@acme.nl /var/tmp/isdnlog
```

The entries define the document to be generated whenever docset mode is executed with the specified *label*. In the example the “daily” label is specified for report generation, which is typically started once every night. The “weekly” label is specified for data export, which may run once a week.

NOTE: labels don’t have a meaning. They are just a name. The “daily” label could run once a week and “weekly” twice a day. The `crontab` entry specifies how often a certain type of docset run is executed. If a new *label* is used (please use “meaningful” labels), a new docset run must be added to the cron subsystem via the static `config.crontab(5)` configuration file or the more dynamic `/usr/local/etc/local/local/config_crontab` script if necessary.

The *docname* field defines the name of the document to use. The *docname* for the report in the example is “http”, which is the ASCII http proxy report. The “*docname*” for the export specification is “auth”, which is to say that all authorization data are to be exported. For exports, the *type* field must be specified. In the example type “csv” is specified: Comma Separated Values.

The input data to be used consists of all data in the `proclog` data area gathered since the last docset run for the specified *label*. A checkpoint is set for a specific *label* for each specific run. All collected data since the last checkpoint is provided to the report program or export generator.

NOTE: the very first time docset mode is started for a certain label, no checkpoint has been set. As a result, a new checkpoint will be created but no output will be generated.

The output from the report program and the export generator is sent to one or more specified *destinations*. The following *destination* types are possible:

email-address

A name with a @ in it is assumed to be an email-address. Output of the report is sent to the specified address in MIME format. Most formats will be sent with MIME-type `text/plain`, but e.g. DBF will be sent with MIME-type `application/x-msexcel`. Mime types are defined in the `proclog` subdirectory `mime`. Files in this directory are named `<type>.mime` where `<type>` is `rpt` for reports or the name of the export type, e.g. `csv`, for data exports.

variable

A name starting with a \$ is assumed to be a variable. In this version, 2 variables are known:

`$mailoper`

`$mailoper` will expand to the value of the `FW_MAILOPER` variable in the `config.root` configuration file.

`$skip`

`$skip` will skip the rest of the current line, e.g.
..... `/var/log/out1 joe@acme.nl $skip jack@acme.nl /tmp/out2`
will only create `/var/log/out1` and send mail to `joe@acme.nl`.

filename

All other names are interpreted as filenames where output should be stored. It is advisable to use absolute paths, because relative paths specify filenames which are relative to the current working directory. Default settings specify that standard reports will be sent to `$mailoper`.

This mode will write warnings and other messages to the logging subsystem.

report mode

```
proclog -r [-ls] [-i infile] [-o output] [-b bday] \  
          [-e eday] [-z lines] [-p period] report-name
```

Report mode will use the intermediate-records to generate ASCII reports for a specific period of time. This is most useful for testing and for generating specific reports from the archived records.

The mandatory argument *report-name* specifies the name of the report to use. If the *-i* option is used, then all records in the specified *infile* will be processed instead of selected records from the proclog data area. The *-o* option writes output to the named *output* destination instead of STDOUT. Surprisingly this can also be an email address!

If no *-i* option is given, data is selected from the proclog data area, according to the *-b*/*-e*/*-p* options. The *-b* option specifies the begin of the report period (default: yesterday 00:00). The *-e* option specifies the end of the report period (default: yesterday 24:00). Both options expect the date to be specified in standard (YYYYMMDD) or Julian (YYYYDDD) format. E.g. *-b 19991210 -e 20000203* specifies the period from 10 dec 1999 00:00 through 3 February 2000 24:00 and *-b 1999344 -e 2000034* does the same.

However, one can also specify the period length with the *-p* option (default: 0 days). The period is given in days and counts forward when *-b* is specified and backward when *-e* is specified. If neither *-b* nor *-e* is specified, *-p* will count backward from yesterday. E.g. *-b 20001122 -p 5* will result in the period from 22 November 2000 00:00 through 27 November 2000 24:00 and *-e 20002711 -p 5* does the same.

The *-z* option can be used to limit (or expand) the number of lines in a report. Default the number of lines in a report is limited to 100 (one hundred).

E.g. `proclog -r -p 8 -o ftp.dat ftp` specifies that a ftp report must be generated from data collected during the last 7 days and that output must be written to a file called `ftp.dat`.

export mode

```
proclog -x [-ls] [-i infile] [-o output] [-b bday] \  
          [-e eday] [-p period] -t type export-name
```

Export mode will use the intermediate-records to export data for a specific period of time. This is useful for testing and for exporting specific data from the archived records.

The options and arguments are exactly the same as described for the “report mode”. The additional `-t` option however is mandatory and specifies the export *type*.

E.g. `proclog -x -p 5 -o tel.dat -t csv telnet` specifies that all telnet specific data from the last 6 days must be exported in CSV format to a file called `tel.dat`.

kill mode

```
proclog -k log|cp|state
```

This switch is just a method for killing (deleting) several types of proclog info. There are 3 possible arguments which specify the info to kill.

log - delete logging

Deletes all proclog specific logging in the proclog package. That is, all logging generated during the latest run with the `-l` flag in `/system/proclog/log/proclog.log` and all saved “exception” logging in `/system/proclog/log/proclog.save`.

cp - delete checkpoints

Deletes all the checkpoints made for the various docset labels, when proclog is started with the `-d` switch. As a result next run for any label will not result in any output, but will force the first checkpoint to be made.

state - delete state-info

Deletes all remembered state-info. This state-info is used in crontab and newsyslog mode, to remember half-parsed sessions, which cross messagesfile boundaries.

EXAMPLES

To convert the messages in some file `msgfile` into intermediate-records and write the output to the file `outfile` use:

```
proclog -m -o outfile msgfile
```

To generate an HTTP proxy report from some file `infile` containing intermediate-records and write the output to the file `outfile` use:

```
proclog -r -i infile -o outfile http
```

To export data in CSV format regarding authentication from some file `infile` containing intermediate-records and mail the output to `jack@acme.nl` use:

```
proclog -x -i infile -o jack@acme.nl -t csv auth
```

To generate an FTP proxy report from data in the `proclog` data-area, starting at 1 Feb 2000, for a period of 8 days on `STDOUT` use:

```
proclog -r -b 20000201 -p 8 ftp
```

FILES

```
/usr/local/etc/local/proclog/proclog  
    the proclog program  
/usr/local/etc/local/proclog/lib/*.pm  
    proclog library  
/usr/local/etc/local/proclog/gen/*.gen  
    proclog export generators  
/usr/local/etc/local/proclog/rpt/*.rpt  
    generic proclog reports  
/usr/local/etc/local/proclog/exp/*.exp  
    generic proclog export definitions  
/usr/local/etc/local/proclog/mod/*.pm  
    generic proclog modules  
/system/proclog/data  
    proclog intermediate data area  
/system/proclog/log  
    proclog logfiles  
/usr/local/etc/local/*/proclog  
    package specific proclog directory
```

SEE ALSO

`config.proclog(5)`, `proclog(7)`.

BUGS

Please report bugs to: <fwsupport@tunix.nl>.

AUTHOR

Copyright 1999-2003 TUNIX Internet Security & Opleidingen

VERSION

Version \$Revision: 1.12 \$ \$Date: 2003/05/16 13:41:22 \$ (\$Name: PROCLOG_02 \$)