POD Translation by *pod2pdf*

ajf@afco.demon.co.uk

book.pod

Table of Contents
book.pod

About Remstats

Remstats is a system of programs to:

- gather data from servers and routers,
- store and maintain the data for long periods,
- produce graphs and web-pages tieing them together, and
- monitor the data for anomalous behavious and issue alerts

It‘s built on *RRDtool|http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool/*. There is a *proto-FAQ|faq*; feel free to contribute. There‘s also a *to-do list|todo* to give some idea what might be coming.

Where to get it

The best available version is 1.0.9b. You can get it at *http://remstats.sourceforge.net/release/src/remstats-1.0.9b.tar.gz|http://remstats.sourceforge.net/release/src/remstats-1.0.9* The current version is 1.0.13a. (This version may not be available yet, as I will push out new documentation before the new release, to make additions/corrections available as soon as possible.) You can get other versions of remstats from the *source archive|http://remstats.sourceforge.net/release/src*. Since almost all of it is written in *perl|http://www.perl.org/* scripts, there is no binary version.

How to get started

First, you should make sure that you have all the *requirements|required*. Then read the *installation docs|install*. Then read the *server installation docs|install-servers*.

Then check out *run-remstats* which runs almost everything else and documents how all the pieces work together.

Thank-you|thanks.

Architecture

This will tell how the major pieces of remstats work together, when I finish it.

A Bit of History

There were several things which decided how remstats would work. One of the major ones was, of course, *RRDtool|http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/*. I wanted to have good graphs, without having to build them from scratch. I liked the side benefit that RRDs do not grow, lessening the file-management required.

I‘ve dealt with various monitoring packages with long-running daemons which are always needing to be restarted, or checked via cron jobs and I decided that remstats wouldn‘t have that kind of problem.

Back in the bad old days, you couldn‘t count on having an SNMP agent available on every platform, so I decided that remstats wouldn‘t require SNMP to work. Now, of course, there is the wonderful *Net-SNMP project|http://net-snmp.sourceforge.net/* to provide a portable SNMP implementation.

I wanted it to be easily extendable, without having to completely understand the guts of remstats. This goal has not been met very successfully, though I hope to improve this in the 2.x releases. However, some folks have managed to create new collectors, so I guess it wasn‘t a complete failure.

The Collection Process

The process of data collection in remstats is begun by the *run-remstats2* program. (In older versions, *run-remstats*, a more primitive program, filled the same niche.) It is started by cron at the minimum collection interval. It passes through several stages in the order listed in the *run config-file|configfile-run*. Some of these stages may have conditions on them. E.G. the standard stage `pagemakers` will only be run if the remstats configuration has changed.

Each stage is described by a file in the *run-stages config-dir|configfile-run-stages*, which lists all the programs to run. These are all started and run in parallel. When all the programs (or pipelines) are finished, T<run-remstats2> moves to the next stage.

The standard stages are:

check - checks the configuration to make sure it‘s good, and as a side-effect creates any required new/missing RRDs, other files or directories.

ping - runs the *ping-collector* to see which hosts are up, so as to avoid wasting time on hosts which are down. Hosts which aren‘t pinged are considered to be "up" for collection purposes, to deal with un-pingable hosts.

collectors - runs all the other *collectors* in parallel

post-collector - runs the *error-collector*, which collects info on the collectors.

monitors - runs the remstats *monitors* which evaluate the current status of the monitored systems and issue alerts, if warranted.

pagemakers - runs a motley crew of programs to create various kinds of web pages, but only if the configuration has changed. One of the standard cron jobs will touch the man configuration directory to force page generation once a day.

Collectors

At the centre of remstats are the *collectors*, and their *servers*. The collectors do whatever they need to do to acquire data, reformat it to the remstats standard collector form and pass it to an instance of the *updater* which does the RRD updating. Some collectors have a corresponding server which they will contact on remote hosts, and which will provide information on request. The servers do not know about the configuration of remstats and are completely stand-alone; the collector will tell them what kind of data it requires and the server will provide it in a standard form.
XXX

Release Notes

Note: These release notes do not cover the usual install, just differences from previous version, new features and special procedures involved in upgrading from one version to another. In other words, this isn‘t a short-cut, it‘s **more** stuff you‘ve got to read. Sorry.

Release Notes for Remstats version 1.0.13a

Bug Fixes

Significant changes

Dropped *graph-writer* from distribution. If you‘ve already moved to using *run_remstats2*, you‘ll already be using *page-writer* instead. If not, switch now.

Release Notes for Remstats version 1.0.12b

Bug Fixes

- Problem with time-sync for remstats servers fixed, see item 198 in *done*. Unless all your hosts running remstats servers are time synchronized, you need this release. I guess that‘s why I never ran into this before. :-)

Changes

- If you‘re upgrading, you should switch from *run-remstats* to *run-remstats2*. The former will be going away soon and the latter is much more flexible.

- moved the default location of the datapage directory. If you don‘t change yours, then all the pagemakers will run every time, which is bad. Edit your `general` config-file and change `datapagedir` to something outside /home/remstats/etc/config. The new default is `/home/remstats/data/DATAPAGES`, which you should probably use unless you have reasons to do otherwise. Then move /home/remstats/etc/config/datapages to /home/remstats/data/DATAPAGES, or wherever you decided to put it.

- replaced the remoteping rrd with remoteping-*, allowing one RRD per *remoteping-server* host.

- renamed `cleanup` to *remstats-cleanup* to avoid name collisions. Make sure that you change the remstats user‘s crontab.

New Features

- new docs section on *troubleshooting*. Check it out and suggest additions.

- *check-rrdlast* will show the last-update times for all RRDs and optionally the data for that last update.

- new *error-collector* to count errors/aborts in the other collectors and thus allow alerts based on this.

- new *alert-syslog* alert-sender to log alerts to syslog

Release Notes for Remstats version 1.0.11b

all configuration files/directories will be checked to see if the configuration has changed, to determine if *run-remstats* needs to run the *pagemakers*.

New *rt-updater* program to watch info on a specified host+RRD+graph in near real-time. It's not well integrated with everything else, but it's still usefull for testing.

New *program-collector* to run external programs to supply values for RRDs, written by Arnauld Michelizza.

All the *collectors* now honor a stop file. If there is a file in the configuration-specified tempdir called `STOP-programname`, then the collector called `programname` will exit after reading its configuration. This is more for debugging, but there may be other reasons to use it. This is similar to the feature in *run-remstats* and *run-remstats2*.

New *show-config.cgi|show-config-cgi* to allow configuration browsing. No editing. Also, it's incomplete, but it does do the important pieces: hosts, and rrds and what they refer to. However, it's not secure, in that it may reveal too much, such as SNMP community strings. Make **very** sure that you control access to it.

`Make install` now will make default htpasswd and htgroup files for use with *apache's|http://httpd.apache.org/* htpasswd feature, if it can find an `htpasswd` program in the path. These will also be used by the new *access_control|configfile-access* routine which will be used to control access to all the *cgis*.

`configure` will now do proper checking for required and optional perl modules. Missing required modules will abort `configure`; missing optional modules will mark the files which require them so that they won't be syntax checked (which would fail).

Release Notes for Remstats version 1.0.10b

If you see this version, you're really fast. Please remove it and take 1.0.11b instead.

Release Notes for Remstats version 1.0.9b

Note: There is a new target in the Makefile: `upgrade`. It's supposed to do the required changes for folks who are upgrading from previous versions. It won't work for upgrading from versions before 1.0.4a. It has **NOT** been exhaustively tested, especially for earlier versions. If the install target notices that the configuration directory exists, it will try to do the upgrade for you. To use it manually:
BACKUP YOUR CURRENT CONFIGURATION!

```
make all
make upgrade
make install
...
```

New Features

The new *run-remstats2* program replaces *run-remstats*. It also brings two more config-files *run|configfile-run* and *run-stages|configfile-run-stages*. This permits `run-remstats2` to be completely configurable in what programs get run. This can be used to run more processes in parallel, but intelligently divided, to permit the run to complete more quickly.

[UPGRADE] If you are upgrading, this will require you to do the following, before doing the install:

```
mkdir /home/remstats/etc/config-base/run-stages
ln -s /home/remstats/etc/config-base/run /home/remstats/etc/config/run
ln -s /home/remstats/etc/config-base/run-stages /home/remstats/etc/con
```

The new `headerfile` directive for the *hosts config-file|configfile-hosts* permits you in inject arbitrary information into the host header.

The new `currentvalue` directive for the *rrds config-dir|configfile-rrds* permits you to store the current value of variables in status files in the host's data directory. Particularly useful in conjunction with the `headerfile` directive. Can also be used to permit other processes to act on values collected by the collectors.

A new section, `sar`, for the *unix-status-server*. This runs "`sar -A`" and parses the result. Unfortunately, there is no consistency about what data is available or when the various items should be called, so the RRDs are OS-specific.

Release Notes for Remstats version 1.0.8a

Another minor release.

New Features

Added –`with-perl=xxx` option to `configure` in case you need a specific version of perl for some reason.

Added `HTMLDIR`, `CGIDIR` and `CGIURL` to the list of *magic cookies|cookies* substituted.

New programs *port-query* and *pattern-query* for testing scripts used by the *port-collector*.

Release Notes for Remstats version 1.0.7a

Another minor release. This one gets rid of GIFs and replaces them by png files.
Also some minor bug-fixes.

Release Notes for Remstats version 1.0.6a

Oops. There was a small glitch in the install for 1.0.5a. Here's a fixed version.

Release Notes for Remstats version 1.0.5a

- [UPGRADE] If you're upgrading from a previous version of remstats, you'll have to:

```
cp /home/remstats/etc/config-base/ntops /home/remstats/etc/config
```

after the install of the new version, even though it's effectively empty and you're not using the new *ntop-collector* (yet). See below for more pointers on it.

- [UPGRADE] You'll need to do the following to accomodate the new *page-writer*:

```
cp /home/remstats/etc/config-base/pages /home/remstats/etc/config/page
mkdir /home/remstats/etc/config/page-templates
cp /home/remstats/etc/config-base/page-templates/* /home/remstats/etc/
```

after the install of the new version.

- [UPGRADE] The host `tags` directive, has been renamed to `keys` to avoid completely confusing me.

- [UPGRADE] The new `environment` config-file requires you to:

```
cp /home/remstats/etc/config-base/environment /home/remstats/etc/confi
```

after the install of the new version.

- The collectors now all will use configuration-supplied IP numbers, if they are there, in preference to looking the host-name up in DNS. If you find an instance of this not happening, it's a bug.

New Features

- RRD config-files can now (like host config-files) have `keys` associated with them. This was mainly added for the new *page-writer*, but could be of other use.

- *graph-writer* has been replaced by *page-writer*, which writes directory trees of pages from simple templates. The original look of the pages graph-writer created has been re-created with 8 templates totalling 94 lines. Other completely different interfaces can now be created.

- *macinfo* a script (and *cgi|macinfo-cgi*) to look up MAC addresses from an ethernet switch via SNMP and show, for each port on the switch, which host is on that port (by MAC address or IP number or domain-name, or a combination). The related script *update-switch-ports* will also update the description of each interface on the switch, with the host which is connected to it.

- The *unix-status-server* has some new sections to make available various data from /proc under linux, including disk I/O, memory, ...

- *alert-monitor* get some new relations: <daystddev, <weekstddev <monthstddev (to compare standard-deviations), | |< (absolute value comparisons), delta>, delta< (change comparisons).

- the *links config-file|configfile-links* now has a more flexible definition and all the links are now configurable.

- the *lockfile* script and *remstats-backup* which uses it.

- the *hosts config-file|configfile-hosts* has a new directive `tags` to allow processing of

sub-sets of all hosts.

- The new *ntop-collector*, configured by the *ntops config-file|configfile-ntops*, allows you to collect stats on protocol usage distribution.

- The *graph.cgi|graph-cgi* script now allows the time-span to be user-specified, and also the graph width and height.

- All the collectors now accept `-H`, `-G` and `-T` flags to allow them to be run on subsets of all hosts, for an explicit list of hosts (`-H`), all hosts in a list of groups (`-G`), or hosts which have a particular tag (`-T`).

- A new *dbi-collector* to collect information from remote databases. Look at the rrd `dbitest1` for an example of how to create your own.

- A new config-file *environment|configfile-environment* to set up the environment for children of run-remstats.

- The *log-server* implements a new function `lastof` to permit it to match patterns, while maintaining state from previous runs. See the RRD `nfsproblems` for an example.

- A new utility *remstats-rrdtune* to tune all the RRD files to match the remstats RRD definitions. Or as much tuning as rrdtool will do. The big one that you probably wanted, mucking with archives, it doesn't do.

Release Notes for Remstats version 1.00a4

[There is no version 1.00a3.]
It's always a good idea to run *check-config* after changing any of the config-files, but it's also a good idea to run it after doing an upgrade, especially when, as in this version, there are changes to the config-files.
Mostly small new features and bug-fixes, except:

- **Incompatible:** re-written alert-sending mechanism. Permits easily written new methods of sending alerts, by separating the alert-text generation (see *alerter*) from the alert-sending (see *alert-email* and *alert-winpopup*). The new *alert-destination-map|configfile-alert-destination-map* config-file permits mapping an alert-destination to different addresses depending on the time-of-day, day-of-week, ... and its alias facility permits sending to a list of addresses which may use different methods of sending the alert.

- **Incompatible:** The *unix-status-server*'s do_df now returns **bytes** not **K-bytes**. This avoids silliness in the graphs saying that you've got 20k kbytes free. It may have been correct, but it wasn't intuitive. You can multiply all the old numbers by 1000 to convert RRDs.

- **Warning:** *new-config* now copies the configuration files which you are likely to change, so that your changes won't be overwritten by an update to remstats. Unfortunately, as all updates (including this one) will overwrite config-base, so people upgrading from a previous version of remstats should convert the following files from symlinks to config-base into copies of those files:

    alerts alert-destination-map general html links tools
    You can do this by running the supplied *convert-config-links* script **BEFORE INSTALLING THIS VERSION**

New Features

- the new *datapage-status* script generates datapages for each host, which will show the current values of all rrd variables. There is a new tool in the default *tools config-file|configfile-tools* which will show this page, and it's been added to the defaults generated by the *new-xxx-hosts|config-tools* programs.

- *Host templates|configfile-host-templates*. So that you can configure similar hosts like:

```
desc     whatever
template their-template
```

Can also be used to make changing some things for many hosts easier. E.G., you could have a template, say `default-nt-status-server` which contained:

```
nt-status-server some-host
```

and configure all hosts which use c<some-host> like:

```
template default-nt-status-server
```

- New *availability-report* and *availability-report.cgi|availability-report-cgi* and config-file *availability|configfile-availability* for reporting on "availability".

- New *nt-status-server* and *nt-status-collector* and RRDs for them (ntactivity, ntmemory, ntpaging, ntnetwork and ntlogicaldisk-*).

- New *cleanup* program to remove stale files.

- New *new-snmp-hosts* now adds other rrds than snmpif-*

- New *new-unix-hosts* program to add hosts which are running the *unix-status-server* with the apropriate rrds.

- ought to work with perl 5.6 now. I'm not using perl 5.6 on the main collector yet, but it seems to install correctly on a test system.

- *run-remstats* now checks all configuration sub-directories to figure out if anything has changed, so you ought to be able to just edit files and the changes will get caught on the next run.

- remstats internal instrumentation allows monitoring remstats collectors, for now. More later. Look at the pseudo-host _remstats_.

- removed old Overall Index, since I never looked at it, and wrote a new RRD Index, which I was wanting.

- new *nt-discover* program to discover and add NT systems

  Note: this adds the new *discovery config-file|configfile-discovery* which must be locally configured. I won't even attempt to guess at values here.

- The old `alertflag` entry in the *html config-file|configfile-html* has been replaced by three new entries: `alertflagcritical`, `alertflagerror` and `alertflagwarn`, allowing e.g. different colors for the different levels of alert.

- *datapage.cgi|datapage-cgi.html* now does variable substitution properly for HTML macros. See the example datapage upss.page under /home/remstats/etc/config/datapages.

- *datapage.cgi|datapage-cgi* and *dataimage.cgi|dataimage-cgi* have two new commands: `alertstatus` and `alertvalue` to fetch alert statuses and values. To be used on forthcoming status pages.

Release Notes for Remstats version 0.13.1

I fixed a minor buglet in 0.13.0 which was noticed shortly after release. I was annoyed enough with it that I made 0.13.1.

Release Notes for Remstats version 0.13.0 (AKA 0.12.2)

There are lots of little improvements, which are detailed in the *Change History|changes*, which I'm not going into here. The main incompatible changes are:

- The configuration structure ($main::config) now has the graphs stored under $main::config{RRD}{$wildrrd}{GRAPH} instead of $main::config{GRAPH}, so there won't be problems with having the same graph-name defined under two different rrds. This will only affect you if you've been writing your own code for remstats, like a new page-maker. I thought that the bug was annoying enough and difficult to figure out when it was triggered that the incompatibility was worth the change.

- After typing $main::config{CUSTOMGRAPH} instead of $main::config{CUSTOM} one too many times, I renamed $main::config{CUSTOM} to $main::config{CUSTOMGRAPH} which is what it should have been all along. Again, this should only affect you if you've been writing your own remstats code, like a new page-maker.

- Changed default location for datapages to /home/remstats/etc/config/datapages, so that all the configuration, including the datapages are together.

- Removed the general config-file directive `pagesas` since all the generated pages are CGIs now. Check-config will abort if you still have it. Just delete that line in the general config-file.

- `use strict` in all the scripts (unless I missed some) in preparation for perl 5.6, which

doesn't like `use vars`. Shouldn't bother you unless you've been writing remstats code, in which case, you probably know what to do.

- To deal with alert templates (see below), you'll need to manually fix your config-dirs. For each one, you need to:

```
su remstats
cd your-config-dir
cp /home/remstats/etc/config-base/alert-template-map .
mkdir alert-templates
cp /home/remstats/etc/config-base/alert-templates/* alert-templates
```

- remoteping-collector has been modified to return the server-name instead of a number to differentiate the data from different servers. There's also a new remoteping-* wildcard RRD to make it more usefull.

Customgraphs on host-index pages

The new `customgraph graphname` directive for host config-files permits you to add a customgraph to a host-index page. (Thanks Marek.)

graph.cgi - remstats graphs anywhere

Like it says, using the new *graph.cgi|graph-cgi*, you can put remstats graphs on any page you want.

Views

You can now define your own pages with page-layout of your choice using *views|configfile-views*. (Thanks to Marek and Thorsten and Matt and anyone else I've forgotten.) Don't forget to add *view-writer* to the list of pagemakers if you've changed the default.

ping-* rrd

You can now ping different interfaces on a host separately (Thanks Steve)

fileage section for unix-status-server

This allows you to fetch the last-modification-time for specified files. It was written to allow remstats to monitor lock-files to check for stale locks. There is no included rrd using it as lock-files are all over the place.

port-collector can collect data from results

The *port-collector* has always been able to send a string to remote services to that it could tell if they were working correctly. Now it can pull values for RRDs and status-files from the results as well. I've included a sample rrd (weathernetwork) and script (weathernetwork) to collect current weather data for Ottawa. Look at the updated docs for *scripts config-files|configfile-scripts*.

new script - snmpif-description-updater

The *snmpif-description-updater* will keep the descriptions on snmpif-* RRDs up-to-date with whatever you've set as ifAlias for that interface. (Thanks Steve Francis.)

Alert Templates

This feature allows you to customize the alert messages by addressee or by RRD. Look at the docs in *alert-template-map|configfile-alert-template-map* and *alert-templates|configfile-alert-templates*.

Autoconf-like configure

You can now do:

```
./configure
make
make install
```

for the beginning of of the *install|install*.

Release Notes for Remstats version 0.12.1

Ideally, this document will only have to tell you about the great new features of remstats in this version.
Not this time.
In addition, due to various stuff (read the *Change History|changes*), this covers changes since version 0.11.1.

Configuration File Replaced by Configuration Directory

The old "one huge configuration file" has been replaced by a directory of files and sub-directories. (See the *new configuration docs|configuration* for details.) This means that most programs don't need to read and parse everything, including stuff that they're not going to use. It also makes it easier to find things, as you can go directly to the file that has what you want, e.g. details on a particular host. It also made possible the newly revamped replacements for `make-ping-hosts`, `make-port-hosts` and `make-snmp-hosts`, which will insert their additions directly into the appropriate configuration files.

There is a new script, *split-config*, which will take your old config-file and a new name and generate a new config-dir from it.

On a related note, I broke the *groups|configfile-groups* line out of the `general` config-file into its own file. It's easier to see what you've got. `split-config` will do this for you. Also the (undocumented) [html] section will absorb large portions of the [general] section which really belong to wep-page generation.

If you've made your own collector, you'd better look at the new skeleton-collector for the required changes. Just change `read_config` to `read_config_dir`, with extra args. There's also documentation on how to write your own *collector|collectors*.

do-remstats replaced by run-remstats

The old `do-remstats` shell-script and all the kludgy shell-scripts that went with it and the `watchdog` and `lockfile` scripts have all gone away. The replacement *run-remstats* does everything they did and does it correctly. It's also configurable, so you don't need to modify the scripts to change which collectors you want to run, e.g.

A new feature of `run-remstats` configurability is that you can have it run the `ping-collector` before everything else and not bother trying hosts that didn't answer it. You can also choose which *collectors*, *monitors* and *pagemakers* to run.

CGI scripts and non-default config-dirs

At the moment, the supplied CGI scripts don't deal with non-default config-dirs. I do consider this to be a problem, but I need to get this release out to deal with other serious installation problems.

You can work-around this by editing the installed CGI scripts and putting in the correct definition for $config_dir, near the top.

plugin-collector is gone

It was an inefficient, difficult-to-configure, kludge and isn't needed anymore with the new run-remstats.

pre-release testing automated

You won't see it, but I hope you'll all notice the improvement in release quality.

New Known Bugs for version 1.0.12b

All the collectors are skipping some of alternate updates. This is because the code to tell if it's time to collect isn't quite right. For now, I've enabled the `-F` for all collectors in *config-base/run-stages/collectors*. If you've made changes or copies of this, you need to make this update, for now.

New Known Bugs for version 1.0.9b

Alert quench (in *alert.cgi|alert-cgi*) doesn't work, on purpose now. It's related to splitting access for the `webgroup` group. I'll make it work again later.

Similarly, the *log-event.cgi|log-event-cgi* CGI script doesn't work now.

Every now and then, I get error messages from the *alert-monitor* saying that there were no destinations for an alert. The message needs to be improved so that people (myself included) can figure out what's wrong and how to fix it.

New Known Bugs for version 1.00a

Alerts in general - The alerts shown by the *alerts.cgi|alerts-cgi* page never get expired. [FIXED <1.0.12b]

If a hub is down, then you'll get alerts for everything behind it. Ought to only get the alert for the hub.

New Known Bugs for version 0.12.2

Neither new-snmp-hosts, nor snmp-collector use get_ifname, with the consequence that neither copes well with "oddly" named interfaces, say with spaces in them. [FIXED in 0.12.3]

New Known Bugs for version 0.12.1

CGI scripts don't work with non-default config-dirs. I consider this a bug, but I need to get this release out now to deal with serious installation problems with previous releases. For now, do the following for each config-dir:

```
% make install-cgis CONFIGDIR=/wherever/you/put/it
```

> Unless someone else suggests an acceptable fix, or I have a an inspiration, this isn't going to be fixed. With both the new *page-writer* and *run-remstats2*, there are fewer reasons to need this. [ACCEPTED 1.0.12b]

run-remstats only checks the config-dir for change, not the subdirectories. For now, just `touch config-dir` whenever you make a change. [FIXED <1.0.12b]

customgraphs are completely broken. Urgh. Upgrade. [FIXED in 0.12.2]

You can't have two graphs of the same name, even in different rrd definitions. This is just flat-out wrong and will be fixed. Unfortunately, the fix will mean even longer file-names, so I hope nobody has some old system with the 14-character limit. [FIXED in 0.12.2]

graphs with descriptions can't have quotes in the description. [FIXED in 0.12.2]

To-Do List for Remstats

Unclassified Priority

High Priority

214 20030516 [HIGH]- finish architecture docs
213 20030515 [HIGH] - change localtime to gmtime througout
212 20030509 [HIGH] - fix *dns-collector* to have the rrd definitions specify the record-types instead of the rrd instances.
210 20030508 [HIGH] - allow configurable wait between items in a run-stage to allow slow machines to cope better (thanks Marek).
211 20030508 [HIGH] - iptables collector or section for unix-status-server Allow configuration of which tables and chains to look at.
208 20030508 [HIGH] - fix docs for new-xxx-hosts to make it clearer that "hostsfile" is just a list of hosts, not a host configuration file.
206 20030328 [HIGH] - fix showlog.cgi - it doesn't like March 29, 30 or 31. Probably related to February.
204 20030128 [HIGH] - only log alerts on transitions Right now, it logs every time after the initial waiting period. It ought to log right away, and not again until there is another change.
202 20020821 [HIGH] - add walk capability to *snmp-collector*, so that it can collect things with multiple instances. The problem is figuring out how to identify the instances. Could be as simple as a wildcard RRD with an enhancement on the `oid` line to allow the wildcard to be propagated to there, if the number is sufficient as an ID. Ideally, I'd prefer a name, but that would require some way of specifying a number-to-name mapping which wouldn't get out-of-date. Many oid trees have a name for the instance as one of the oids. [Ecaroh] See also **42**.
200 20020819 [HIGH] - monitoring of remstats errors - While cleaning up stderr files, count errors and aborts and log that too.
- [DONE 20020820] - make sure it works.
193 20020726 [HIGH,BUG] - look at datastuff.pl RRDNODATA causes all sorts of error messages in the web error log if it gets RRDNODATA.
191 20020716 [HIGH,BUG] - *alert-monitor* is logging too much It needs to only log state transitions.
187) 20020627 [HIGH] - some df's give blocks instead of k
185) 20020625 [HIGH,BUG] - host-templates are broken In some cases they wipe out all RRDs from $main::config. Unfortunately, I can't duplicate this any more. Double-plus-ungood.
99) 20000619 [HIGH] make unix-status-collector send the directories that we want df for and make unix-status-server do "df /dir1 /dir2" to get them, and pull them off one line at at time. This is to deal with things like disconnected NFS-mounted directories hanging df when we do just a bare "df".

86) 20000419 [HIGH] trends analysis
- [20020621] consider Holt-Winters smoothing to create temporary RRD files to show smoothed data and projections. Store it under GRAPHS/TMP
87) 20000419 [HIGH] alerts based on trends analysis and historical data, like one-week average and standard-deviation, ... (for Steve)
- [20020621] new kind of alert using Holt-Winters to extrapolate trends and base the alert on some value in the future. Definition like:

```
alert trend(xyzzy,7d) < 20 40 60
```

or maybe

trend-alert xyzzy 7d < 20 40 60

to trigger an alert if the trend for variable `xyzzy` at `7 days from now` is ... (the rest interpreted in the usual alert manner.)
Could even generalize this by making it `&trend` instead of `trend`, which would apply the named function to the data. Probably not worth implementing, as I can't think of what I'd do with it.
146) 20011220 [HIGH] - deal with broken html in release/src directory

---

Medium Priority

215 20030516 [MED]- make *nt-status-server* not need srvinfo, as it can be really slow.
207 20030404 [MEDIUM] - alert-fixup - run command on detecting condition The command is found in `/home/remstats/etc/config/fixups` and is named either `address/hostname/rrdname/varname`, `address/ANY/rrdname/varname`, `address/ANY`, `ANY/hostname/rrdname/varname`, `ANY/ANY/rrdname/varname`, `NOMATCH`, or `ALL`looked for in that order. The default, `NOMATCH` will be supplied with remstats distribution and will do nothing, but may be altered to act on all non-matching fixups. The others are site-specific. `ALL`, if present, will be run after any specific script is found. Scripts will be run with an environment consisting of all the magic cookies which can be substituted into a regular alert message.
203) 20030122 [MEDIUM] - move host data to data/HOSTS/<hostname>
188) 20020628 [MEDIUM] - new show-alert-thresholds.cgi to extract the levels from rrds and host (separately) and format them prettily.
180) 20020613 [MEDIUM] - development hints E.G. use skeleton-collector as template to make new collectors.
168) 20020508 [MEDIUM] - check that we're not lying with /o in pattern matches.
127) 20010622 [MEDIUM] - graph data together with historical data. This will probably mean either populating another rrd with historical averages, temporarily or permanently, or modifying rrdtool. The former is certainly simpler to do, given my knowledge of the internals of rrdtool. However, it needs to have another rrd for each period? Need to keep the same data over some longer period, a multiple of the period of interest, as well as the averages, from period to period.
- see also 86 and 87.
138) 20011002 [MEDIUM] - make new-config symlink all rrds from config-base instead of just the directory. That will make it easier for people to have their own rrds which don't get overwritten.
137) 20011002 [MEDIUM] - make datapage-interfaces include if-* and procnetdev-* collected interfaces as similar info is available.
131) 20010824 [MEDIUM] - make status pages for each host, group and for all hosts using the new alertstatus and possibly alertvalue.
- [DONE before 20011220] see *datapage-alert-writer*
109) 20001212 [MEDIUM] `nt-log-collector`, with modules for event-logs and ordinary log-files. Note that this implies a new `nt-log-server`. Or it could be added to the *nt-status-server*. Probably should be a new service as the `nt-status-server` can already be quite slow.
106) 20000922 [MEDIUM] make a file-collector. Similar to the log-collector, only for small, local files. Slurp the file into memory, match patterns and pull out values. The data line in an rrd definition would be like:

```
source file
data VARNAME    GAUGE:600:0:U FUNCTION PATTERN(WITH)PARENS
```

In fact, this would share so much code with the log-collector that it might be worth combining the two. This allows collection from things like Linux's /proc.

103) 20000915 [MEDIUM] make-path doesn't work with non fqdn hosts. Make it read the configuration, so it can look up the IP number in the host config and use that if it's defined. Otherwise, default to gethostbyaddr.
45) 20000121 [MEDIUM] make snmp-collector send only one packet per host
- test and make sure that we do get back whatever succeeded. I vaguely remember that it didn't work. [Later: at least under UCD snmp under linux, if an item isn't implemented in the MIB, you get back NOTHING. Specifically, look for the non-unicast packet counters as well as something else; you get nothing back. This isn't good.]
- have to re-write snmp-collector completely, which isn't that bad an idea. This means a two-pass structure. On pass one, we construct the complete query and then send it. On pass two, we examine all the results and format them.
9) ???????? [MEDIUM,TESTING] make alerts take connectivity dependence into account
- add "via" line to host section to deal with hubs and switches [DONE]
- I think it's done. See what happens next outage.

Low Priority

134) 20010829 [LOW] - make header_bar (in htmlstuff) do the link making, if available and fix whatever uses it not to.
133) 20010829 [LOW] - add an option to make nt-discover update old hosts with a standard set of RRDs, even if the hosts are already known.
102) 20000912 [LOW] add see-also to host config, which will materialize links in the host header. Config line like:

```
seealso host:xyzzy http://www.somewhere ftp://ftphost
```

the special "host:" pseudo-URL gets changed to a link to the remstats page for that host.
51) 20000216 [LOW] need a way to specify URL for port-http. The root page doesn't always exist.
37) 19991216 [LOW] traceroute sometimes shows incorrect routing, which confuses the topology-monitor, causing false positives
50) 20000215 [LOW] make inventory script. Runs uname (for hardware and software), `ifconfig -a`, `netstat -nr`, `hostname` and any others I can think of to collect configuration info. Then figures out the versions of important software, e.g. run `perl -v`, `gcc -v ...` Make a subdir to put it in and make a tool definition to get it onto the host pages.
- looks like the beginning of a discovery script.
69) 20000406 [LOW] is there any use for write_environment in check-config?

On Hold

Usually waiting for next major release, or trapped by something else. (in priority order)

High Priority

164) 20020409 [HIGH,HOLDRELEASE,NEEDS=165] - document perl libraries
165) 20020409 [HIGH,HOLDRELEASE] - convert perl libraries to modules that can be `use`'d. Consider OO, if there's a benefit which outweighs having to re-write almost everything.
92) 20000518 [HIGH,HOLD] collect traffic info from cflowd (artsportms). Make it flexible enough that it can let you choose which ports you want (one per rrd?). Make a loader to load historical data.
- [DONE 20000524] artsportms-loader done
- [HOLD] I no longer have access to devices with this feature
70) 20000407 [HIGH,HOLD] CGI scripts need to have a way to deal with alternate config-files, and graph-writer needs to tell them if they can't work it out themselves. Otherwise, people need to be told to do multiple installs of the CGI scripts, which might be the best way.

```
make install-cgis CONFIGDIR=config-xxx
```

Not that painfull, but wastefull and makes upgrade messier.
- I don't like the multiple-install method, but any other method needs a way of getting configuration information into the CGI scripts. Any method which passes info in via the URL or form fields is out: too unsafe. The only other method I can think of is to read a configuration file in the same directory as the CGI script. This ought to be safe from modification, or your web-site is waiting to be mutilated. The other part to consider is whether any part of the info in the CGI config-file is sensitive. I.E. do we

have to protect it in some way.
- Configuration file in the same directory won't work either, you'd still have to install the cgi's multiple times. I'm starting to think that multiple installations may be the only safe thing to do.

Medium Priority

60) 20000328 [MEDIUM,HOLD] replace route-collector with something which scales. SNMPwalking bgp4PathAttrBest doesn't scale to large Internet routers with 400 peers, taking over an hour to complete. (see also 61)
- look at a script to follow the output of zebra. That's a lot of overhead though. Easy if zebra is solid.
- How difficult can it be to make a native BGP listener? I'm not clear on the protocol, but it doesn't look too bad.
- [HOLD] As I don't need it, and have no access to anything which does.

42) 20000114 [MEDIUM,HOLD] snmp-collector mod to allow summary data collected from a walk and then filtered as a single data-point. E.G. specify a rrd "oid" like:

```
walk    count ifOperStatus = 1
```

would produce a count of the number of interfaces on that device that were active (i.e. had a live device plugged into them). Or a similar one would let you count BGP routes, or arp addresses, ...
- Unfortunately, from experience with the snmp-route-collector, this is going to be slow for anything with a large number of items.
- [HOLD] Until I think of something to use it for.

171) 20020510 [MEDIUM,HOLDRELEASE] - have collectors log how many hosts, unique rrds and rrd instances they collected from for _remstats_.

121) 20010202 [MEDIUM,HOLD] - how about an discovery program, to find and identify hosts and then run the appropriate new-xxx-hosts scripts to add them?
- DONE 20010608 - nt-discover to find and add NT boxen

128) 20010629 [MEDIUM,HOLD] - custom, configuration-supplied info per rrd which is simply available wherever it makes sense, e.g. in alerts.
- first make sure someone has a use for it.

40) 20000104 [MEDIUM,HOLD] consider some form of access-control for servers
- hash-based "password"
- ssl tunneling ought to work for everything except SNMP
- what does this buy? With the various servers run under tcp_wrappers an attacker must either gain access to the remstats collector machine or spoof a tcp session from them. If you've been "owned" you've got bigger problems. If the attacker spoofs a session with a remstats server, tcp-wrappers will insist that it must come from one of the allowed hosts, so that's where the stolen output will go. This is only usefull to the attacker if they have access to the remstats collector machine or if they can sniff the traffic between the collector and the server. The only data loss possible is with the *log-server* which keeps state. (Ignoring DOS attacks which are always a problem.)
- unless someone needs this, it's on hold

Low Priority

10) ???????? [LOW,INPROGRESS,HOLD] make graph of connectivity

13) ???????? [LOW,INPROGRESS,HOLD] snmp trap listener to update status files
- needs filter to be usefull [DONE]
- I haven't seen any useful traps so this is on hold.

39) ???????? [LOW,HOLD] make RRD dumper, to put data out in a form that can be loaded into a database
- I don't need it, per se, but it might be easier than writing the availability report generator.

52) 20000215 [LOW,HOLD] make a makegraph.cgi, or whatever, that will let you make a somewhat custom graph on the fly. makegraph.cgi by itself will list all the hosts and let you choose one. makegraph.cgi?host=xxx will list all the RRDs for this host and let you choose ?one?. makegraph.cgi?host=xxx&rrd=yyy will list the various DSs for this RRD and let you choose the ones you want. Then you get to define any CDEFs needed and then LINEn/AREA/STACK for each DEF or CDEF desired. And size, title, legends...

- On hold since *graph.cgi|graph-cgi* will let you get at any existing graph you want. If I find a use or need for this, I'll re-activate it.
- see XXX

112) 20001212 [LOW,HOLD] - web-based remstats configurator. Needs to consider security, at least from the point of view that you don't want to lose your configuration. The most important part is hosts. A lot of the rest doesn't have to be changed, or only once.

111) 20001212 [LOW,HOLD] consider grafting on (at least links to) some kind of system configuration interface. For configuring the mmonitored entities, not remstats.

110) 20001212 [LOW,HOLD] consider problem-fixing interface. It'd be nice to try to fix things if there is a known way to do so. A simple kludge would be to add another method to the alert-destination-map which deals with problems that it knows about, possibly invoking plugins for specific alerts.

130 20010823 [LOW,HOLD] - add an <RRD::EXEC ...> tag to rrgcgi.

- [HOLD] I thought I had a use for it, but I can't think of one now.

162) 20020327 [LOW,HOLD] - allow logging as a possible type of alert "notification".

- [HOLD] Why?

147) 20011220 [LOW,HOLD] - get list of links to users

- [HOLD] I don't know what this is.

I've also kept the stuff that used to be here, but has already been *done*.

FAQ for remstats

1 What is remstats?

Remstats is a system of programs to:

gather network accessible data via SNMP and its own servers,
store and maintain the data for long periods,
produce graphs and web-pages tieing them together
monitor the data for anomalous behaviour and issue alerts

2 Where can I get it?

The latest stable version is 1.0.9b. You can get it at *http://remstats.sourceforge.net/release/src/remstats-1.0.9b.tar.gz*. The current development version is 1.0.13a. You can find it with all the other versions in the *source archive|http://remstats.sourceforge.net/release/src*. As most of it is perl scripts, there is no binary version.

3 How do I ...?

Just a sec, there's lots of documentation online on how to set this up. You found the FAQ and that's good, but in order to get remstats installed and configured, you're going to have to read at least the *prerequisites|required*, and *installation|install* sections and all their sub-sections. Go do that now before bothering to read the rest of this FAQ. You'll need to do it anyway and it ought to answer most of the questions about getting going.

Complain|mailto:terskine@users.sourceforge.net if it doesn't.

If you'd prefer to print something and read it offline, you can look at *book.ps|book.ps* or *book.pdf|book.pdf*. Note: The pdf translation is **really** slow loading. Be patient.

We now return you to your regularly scheduled FAQ.

Miscellaneous

This is only a proto-faq, covering the things I couldn't figure out where else to put.

1 snmp-collector complains that I don't have SNMP_Session installed, but I don't want SNMP. Do I have to collect and install it?

No you don't. If you're still using the old *run-remstats*, modify the `collectors` line in the *general|configfile-general* config-file to exclude snmp. If you're using *run-remstats2*, just change the *run-stages|configfile-run-stages* collectors config-file to comment out the the lines for the various SNMP-based collectors, primarily *snmp-collector*.

2 I modified the RRD definition, adding a new RRA, but remstats is ignoring it. How do I

do this?

Sorry. At the moment, remstats won't propagate any changes to the RRD structure after creation. Some changes, like extending RRAs and changing min/max for DSs can be done manually with rrdtool. If you do use rrdtool manually, I recommend that you modify the rrd definition as well, to keep them in sync. At some point in the future, I'd like to try to do this kind of update, and it's more likely to succeed if remstats' rrd definition matches what's in the actual RRD.

3 **How do I move the remstats installation to another directory or machine?**

The best way to do this is to a complete new installation in the new location and then copy the `config` and `data` directories to the new installation. As usual, make sure that the files and directories end up owned by the *remstats user|install-user*.

Documentation Conventions

The only documentation conventions the reader has to know about are:

things inside [square brackets] are optional

parenthesized lists with the items separated by vertical bars, (like | this | one) require that you choose one and only one of the alternatives.

Everything else ought to be explicit. If it isn't, or if you don't understand it, please bring it to the author's attention, stating which part you don't understand. There's not a lot of point in my writing documentation which no-one else can understand. I'd rather do it right.

What you need to install remstats

1 You'll need *perl|http://www.perl.org/*, at least version 5.005_03. If you don't already have it you can get it from *CPAN|ftp://ftp.crc.ca/pub/packages/lang/perl/CPAN/src/stable.tar.gz* (the Comprehensive Perl Archive Network).

2 You'll need a `C` compiler that works. :-) gcc or *egcs|ftp://ftp.crc.ca/pub/packages/egcs/* will do fine and you can find them easily in many different places.

3 Make sure you have the following perl modules installed (most of which you can find at *CPAN|http://www.cpan.org/modules/by-module*). The versions are the versions I'm using, but more recent versions should work too, unless there have been radical changes. They should be installed in the listed order to avoid dependency problems:

RRDs 1.0.33|http://remstats.sourceforge.net/release//src/rrdtool-1.0.33.tar.gz - the key piece. It comes with RRDtool and does the database and graphs. Originally from *http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool|http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool*.

Socket - should be standard if you've got the required perl

IO::Socket - should also be standard in the required version of perl

Time::HiRes 01.20|http:src/Time-HiRes-01.20.tar.gz - used by the *port-collector* and the *dbi-collector* and probably a few others, to determine response time. You can comment out the `"use Time::HiRes"` line and the programs will still work, but the response-time resolution will be one second instead of one milli-second. Originally from *CPAN|ftp://ftp.crc.ca/pub/packages/lang/perl/CPAN/modules/by-module/Time*.

SNMP_Session 0.83|http:src/SNMP_Session-0.83.tar.gz - used by the snmp-collector and a few other SNMP-related tools. If you don't need SNMP, you can leave it out, but you'll have to change the *general config file|configfile-general*. (Modify the `collectors` line to leave out `snmp`.) Or, if you're using *run-remstats2*, you'll have to change the *run-stages|configfile-run-stages*/collectors configfile. Originally from *ftp://ftp.switch.ch/software/sources/network/snmp/perl/|ftp://ftp.switch.ch/software/sources/network/snmp/p*

GD|http:src/GD-1.30.pm.tar.gz - used only by *dataimage.cgi|dataimage-cgi* to create images on the fly. Originally from *http://stein.cshl.org/WWW/software/GD/GD.html*.

Win32::Daemon|http://www.roth.net/perl/packages/ - Used only by the *nt-status-server* so that it can run as a Win32 service.

Win32::PerfLib - Used only by the *nt-status-server* to obtain performance counters.

`DBI` - used only by the *dbi-collector*

`Pod::Pdf` - used only by *podpdf* to create book.pdf. Only needed by people who are changing the remstats documentation, not just using it.

`Time::Local` - used by *availability-report*, *availability-report.cgi|availability-report-cgi*, *nt-status-server*, and *showlog.cgi*, but it ought to be part of the required version of perl.

`Net::YahooMessenger` and `Crypt::PasswdMD5` - used by the *alert-yahoo* alerting method.

4 You'll also need the following programs for the *unix-status-server*. (You can change the locations at the top of it.) You almost certainly have most of these and can ignore any that you don't tell the `unix-status-collector` to query. For details, look in the *unix-status-server* docs:

```
uname vmstat df uptime netstat ps ftpcount qmail-qstat
```

How to install remstats

READ THE *RELEASE NOTES|releasenotes* FIRST. This page is generic and does **NOT** include version-specific instructions.
I know that this is not simple. I do plan to make it simpler, but it'll **never** be "`./configure; make; make install`" because I don't know what you want to monitor.
The two C programs (*multiping* and *traceroute*) now use autoconf, and the main configure script works (from the outside) simlarly to an autoconf-generated configure. I haven't seen a need to convert it to autoconf yet. It's mostly perl scripts and if you have the right version of perl properly installed, it shouldn't need anything special. The `unix-status-server` is a slight exception to this, but the only configuration needed so far is done dynamically and is only the location of the various required utilities.

1 Unpack the distribution tarball:

```
gunzip -dc remstats.tar.gz | tar xf -
```

2 Create the remstats user and group, if you haven't already, (by default `remstats` and `remstats` respectively.) (See also *the remstats user|install-user*.) Unless you plan to run a separate instance of the webserver for remstats, you'll also want to create the *webgroup* group.

3 Build and install the software. If you're upgrading, you might want to take a copy of fixup.config from the old version:

```
sh configure
```

If you want to override the defaults, then run

```
sh configure --help
```

for a list of what can be overridden.
[Check fixup.config to make sure it is properly setup.]

```
make all
make upgrade
make install
su - 'make install-suid'
```

Note: this step also customizes the programs and documentation with your choice of directories, owner, ... so this documentation should refer to your setup after you've done the install.
The `make install-suid` simply makes traceroute and multiping suid root. They won't work most places unless run as root, one way or another. Since I don't like to run all of remstats as root, this was the best compromise I could come up with.

4 Fix the config-base for site-specific things. Edit the following files in /home/remstats/etc/config-base, looking the the string "FIXME", without the "quotes".

```
alerts general html scripts/http-proxy
```

I'll try to keep this list up to date, but you can make sure by doing:

```
grep -l FIXME /home/remstats/etc/config-base/* /home/remstats/etc/config-base/*/*
```

5 Make a *config-dir|configuration* to describe what you want to monitor. You can do this by hand, or using the configuration building tools. To use the tools, you'll have to make a few files listing various kinds of hosts:

```
cd /home/remstats/etc
/home/remstats/bin/new-config config
/home/remstats/bin/new-ping-hosts groupname1 group1-hosts-file
/home/remstats/bin/new-ping-hosts groupname2 group2-hosts-file
...
/home/remstats/bin/new-port-hosts groupname3 port-hosts-file
/home/remstats/bin/new-snmp-hosts groupname4 SNMP-community-string snm
```

After you've installed the *unix-status-server* on some hosts, you can also use:

```
/home/remstats/bin/new-unix-hosts groupname5 unix-hosts-file
```

If you have any Windows NT hosts that you want to monitor, after you have installed the *nt-status-server*, you can run *nt-discover* to find and add the NT hosts for a given NT domain.
If you're going to use the log-collector, you'll have to build the rrd entries for each by hand. There doesn't seem to be much standard in where log-files go, let alone what's in them.

6 Arrange for cron to run *run-remstats2* at an appropriate interval. For a five-minute interval, something like the following will do:

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /home/remstats/bin/run-remst
```

This checks the configuration, collects the new data, updates the rrds, runs the monitors to compute statuses and updates the web-pages. Note: it does **not** re-write the web-pages for every iteration; it only does so when configuration files have changed, as the web-pages will show new data by themselves. This and more are in the sample `crontab.new` file under `/home/remstats/etc`.

7 [optional] Arrange for cron to run `do-traceroutes` at an appropriate interval. You could run it in the wee hours of each morning like:

```
5 3 * * * /home/remstats/bin/do-traceroutes
```

This information isn't currently used, but I'm planning to make use of it.

8 [optional] Arrange for cron to run *snmpif-description-updater* periodically, if you have any snmpif-* RRDs, which you're likely to change the descriptions on. Say every day, like:

```
0 3 * * * /home/remstats/bin/snmpif-description-updater
```

9 Arrange for cron to run *remstats-cleanup* every now and then to remove old un-needed files, like:

```
0 3 * * * /home/remstats/bin/remstats-cleanup
```

This removes no-longer-needed files, like old host graphs, traceroute results, log-files, ... It does **not** remove rrd and status files, on the principle that you might want them and they can't be re-created.

10 You'll need to set up your *web-server|install-webserver* to allow CGI scripts in the remstats html tree and make sure that you're not allowing everyone in.

11 Make a symlink in the html directory from whichever index you prefer to index.cgi.

12 You'll want to look at the *server installation docs|install-servers* if you're going to be running any of the remote servers (*log-server*, *nt-status-server*, *remoteping-server*, and *unix-status-server*).

Enjoy your pretty pictures and I hope that you find them usefull.

The remstats user

You **must** choose a userid to run the remstats processes under. By default, it will be the user `remstats`, but you'll have to create it manually, as I'm not going to risk damaging someone's /etc/passwd file. Many operating-systems have a script called `useradd` or `adduser` or some variant

on that.

NOTE: Don‘t run the remstats programs except as the remstats users. Many of the programs write extra files you won‘t know about unless you read the source, and when you do run them as the remstats user, it won‘t be able to modify the files that were created by the other user. This will probably cause the program to die, with a meaningful error message I hope, and you‘ll have to modify the owner by hand, as root. If you need to do this, go back to the source directory and do:

```
% su - 'make install-owner'
```

The remstats user must be able to write files within the remstats directory trees rooted at /home/remstats, /home/remstats/data and /home/remstats/html. The collection/update processes will also create files under /home/remstats/tmp and /home/remstats/data. The pagemakers write files under /home/remstats/html. It‘s simplest to have all the remstats files and directories(except *multiping* and *traceroute*) owned by the remstats user.

You must also ensure that the CGI scripts (and almost every web-page remstats creates is a CGI script) get run by the remstats user. The CGI scripts read files under /home/remstats/data and /home/remstats/datapage. (See also *the web-server installation|install-webserver*.

There is a recently added feature in `configure` which permits you to specify a `webgroup`, which is the group the web-server runs as. If you are using this feature, you need to add the remstats user to the `webgroup` group and logout and back in so that the remstats user has access to this group to `chgroup` appropriate directory trees, primarily images, that get written as the web-server.

The Webgroup Group

This group isn‘t required if you‘re going to run a separate instance of the web-server, running under the remstats user id. However, if you want to use an existing web-server and just tell it to serve the remstats pages as well, it‘s the easiest clean way to set things up.

The reason it is required is that the graphs images are written to files at run-time. Run-time means here that they are written by whichever user the web-server is running as, often something like ‘httpd’. The remstats user needs to be able to re-write the web-pages and create directory trees for the pages and images. In fact, it needs to be able to create the directories that the images go into.

So we create a ‘webgroup’ group. Let‘s call it ‘xxx’ for this explanation. This group has two members: the web-server user and the remstats user, so the line in /etc/groups might look like:

```
xxx:x:219:httpd,remstats
```

When the *page-writer* creates the directory tree for the graph images (called GRAPHS), it creates the new directories with mode 2775 and group ‘webgroup’ (‘xxx’ in our example). This makes the directories writable by the CGI scripts running under the web-server, and the set-group-id bit will cause the group to be propagated to files created under this directory on most unix variants.

Magic Cookies

There are various places in the configuration file where you can have text substituted for you. It‘s a (very-limited) macro facility. Currently, it only works in graphs, scripts and tools. The cookies are always UPPERCASE and the name is surrounded by "##", so that a request for "color1" would look like "##COLOR1##", without the quotes.

Here they are:

Colours:

You‘re not required to use these, but you‘ll really regret not doing so when you decide to change colors later.

COLOR1, COLOR2, ... COLOR6 and also COLOR1a, ... - generic colours for graphs

PROBLEMCOLOR - an alarming colour

TOTALCOLOR - the full amount of something

USEDCOLOR - how much is used of something

Other stuff:

DB - the full path and file-name of the current rrd

CGIDIR - where the CGI programs live

CGIURL - where the CGI programs live in web-space

DATADIR - where the data files live

GRAPH - the name of the current graph

GRAPHTIME - the name of the current time-span

HOST - the name of this host

HOSTDATADIR - where the current host's data files live

HOSTDESC - the description of this host

HTMLDIR - where the html and graphs live

HTMLURL - where the html and graphs live in web-space

IP - the IP number of the host

IPORHOST - the IP number of the host, if it's defined in the host's config-file, or else its name.

RRD - the name of the current RRD, without the .rrd extension or file-name fixing

RRDDESC - the description of the current RRD

SHORTHOST - the name of the host before the first dot, unless it is a generic name like www or ftp or mail, in which case we use the next part unconditionally.

THUMBHEIGHT, THUMBWIDTH - how large to ask for the thumbnail to be. Not how large the resulting gif is.

WEBMASTER - who is responsible for this web presense

WILDPART - the instance part of a wild rrd. If if-* is being used by if-le0, then WILDPART will be le0.

private.pl - configuration-supplied functions

There are several places where you can have functions you choose invoked by remstats:

- *updater* permits functions to be applied to incoming data via the *rrd definition|configfile-rrds*.
- *datapage.cgi|datapage-cgi* and *dataimage.cgi|dataimage-cgi* permit functions to be applied on any `eval` line.

Rather than me continually adding functions or you having to hand-modify your copy of remstats whenever I make new releases, I've supplied an almost empty perl file called `private.pl` which will be installed in the /home/remstats/lib if you don't have one, but will never be modified by me after that.

Server Installation

One of the interesting things about remstats (I think), is the *remote servers|servers*. To install them, you'll need to do a few things on each host which will run the servers:

Add entries for the servers in `/etc/services`, like this:

```
unix-status     1957/tcp        # remstats unix-status server
log-server      1958/tcp        # remstats log server
```

You can run them on different ports, but these are the defaults and you'd have to change *run-remstats* to add the appropriate switches.

[Optional] Unless you're going to run the servers as root (unnecessary), you'll need to create the user that the servers will run as. The only reasons I can think of for running the servers as root are if you need to run them on a low-numbered port (<=1024), or if you need to read a log-file which isn't readable by the remstats user, or if you want to run multiping non-suid. On Linux and Solaris, you can do:

```
groupadd remstats
useradd -g remstats -d /home/remstats remstats
```

Modify `/etc/inetd.conf` to get the servers invoked, like this:

```
unix-status     stream tcp nowait remstats /home/remstats/unix-status-
log-server      stream tcp nowait remstats /home/remstats/log-server l
```

Or if you're using *tcp_wrappers|ftp://ftp.porcupine.org/pub/security/tcp_wrappers_7.6.BLURB*, which you should be:

```
unix-status  stream tcp nowait remstats /path/to/tcpd /home/remstats/unix-status-server
log-server   stream tcp nowait remstats /path/to/tcpd /home/remstats/log-server logfile1 logfile2
```

And remember to update `/etc/hosts.allow` to allow your remstats host access.

```
unix-status:remstats-host-name-here:allow
log-server:remstats-host-name-here:allow
```

Tell inetd to re-read it's config-file:

```
kill -HUP pid-of-inetd
```

copy the remstats servers to the machines which will run them

```
scp unix-status-server log-server remoteping-server multiping host:/ho
```

The nt-status-server

This one is a bit different to install. I've only done it under the *ActiveState Perl|http://www.activestate.com/Products/ActivePerl/index.html* under Windows NT 4.0. Installing the ActiveState Perl is straightforward; if you got here, you'll have no trouble with that. Installing it as a service is not as simple as I intended. You'll have to get the SRVANY and INSTSRV programs from the NT Resource Kit, and follow their instructions. The program that SRVANY will be running is, of course, perl (usually `C:\Perl\bin\perl.exe`), and the argument string something like:

```
c:\wherever\you\put\nt-status-server -s -t 10.111.12.13
```

You'll have to replace `c:\wherever\you\put` with the path to *nt-status-server*, and `10.111.12.13` with the IP number of the host running the *nt-status-collector*.
Or you can run it not as a service like:

```
c:\wherever\you\put\nt-status-server -s -t 10.111.12.13 -s -T
```

but you'll have to start and stop it by hand at the console.
It also needs the `Win32::Daemon` module, as noted on the *required|requirements* page. The simplest way to install this (under ActiveState), is to use the `PPM` program that came with ActiveState Perl. Like this:

```
c:\> ppm
ppm> set repository roth http://www.roth.net/perl/packages/
ppm> install Win32::Daemon
ppm> exit
```

Currently, the code needs this module, even if it's invoked with the `-s` flag, which doesn't use the `Win32::Daemon` code at all.

Getting your web-server ready for remstats

Choosing userid for remstats

Almost all the remstats web-pages are generated by some kind of CGI script. Many of them will read additional files not available under the html directory tree. In order to provide access to these files, the simplest way is to make sure that the scripts get run as the remstats user. You could also accomplish this by making the web-server user a member of the remstats group. The simplest way to make sure that the scripts can read all the files they need to, is to run a separate instance of the web-server software as the remstats user. You may have other methods of accomplishing this, depending on the web-server you're using. (See also *remstats user|install-user*.) Note that some scripts also require write access, specifically *datapage.cgi|datapage-cgi* (sometimes), *alert.cgi|alert-cgi* (sometimes) and *log-event.cgi|log-event-cgi* (always).

Running CGI scripts under the remstats tree

You also may need to tell your web-server that `xxx.cgi` means that this file is a CGI script and needs to be run, instead of just displayed. With the *apache|http://httpd.apache.org* web-server, you could add the following lines to the `httpd.conf` file:

```
<Directory /home/remstats/html>
Options FollowSymlinks ExecCGI
AddHandler cgi-script .cgi
</Directory>
```

and

```
ScriptAlias /remstats/cgi-bin /home/remstats/cgi-bin
<Directory /home/remstats/cgi-bin>
Options FollowSymlinks ExecCGI
</Directory>
```

Restricting access to CGI scripts

[You should also look at the *access config-file|configfile-access*.]

There are a few things you should do before telling others about remstats. Remstats comes with a few CGI scripts which you probably don‘t want to make publicly available and two that you certainly don‘t. `ping.cgi`, `traceroute.cgi` and `whois.cgi` should probably be restricted to your own organization, unless you don‘t mind letting anyone on the Internet run pings, traceroutes and whois queries from your domain. Rectricted to your domain, you only have to worry about your own people.

However, `alert.cgi` and `log-event.cgi` are a different kettle of fish. They will permit anyone who can run it to quench alerts and log comments about them. You will probably want to be a bit more restrictive about who you let run this.

Using the *apache|http://httpd.apache.org/* web-server, you can restrict the use of these CGIs using a `.htaccess` file something like this:

```
# Note that this example uses the private network 192.168.0.0.
# Stuff to make Apache expire the files to get them refreshed
ExpiresActive on
# images every 5 minutes, when the data gets updated
ExpiresByType image/gif M300
ExpiresByType image/png M300
# html every day
ExpiresByType text/html M300

# What to allow
Options ExecCGI FollowSymlinks Indexes

<Files "^(whois.cgi|traceroute.cgi|ping.cgi)$">
order deny,allow
deny from all
allow from 192.168. 127.0.0.1
</Files>

<Files "^(alert.cgi|log-event.cgi)$">
order deny,allow
deny from all
allow from 192.168.20.1 192.168.23.3
</Files>

# How they’re allowed in
order deny,allow
allow from all
```

I won‘t claim the IP#-based access-control is completely safe, but it‘s easy and keeps out casual browsers. If you **really** need to keep this information safe, use a secure web-server, say apache with mod_ssl. If that‘s not good enough, you ought to consider whether this stuff really belongs on a network at all.

The Configuration Directory

The run-time configuration of remstats is done through a directory-tree of files. The current tree structure is:

```
configdir
    +--- L<access|configfile-access>
    +--- L<alert-template-map|configfile-alert-template-map>
    +--- L<alert-destination-map|configfile-alert-destination-map>
    +--- L<alert-templates|configfile-alert-templates>
    |       +--- template1
```

```
|       +--- template2
|       ...
+--- L<alerts|configfile-alerts>
+--- L<archives|configfile-archives>
+--- L<availability|configfile-availability>
+--- L<colors|configfile-colors>
+--- L<customgraphs|configfile-customgraphs>
|       +--- graph1
|       +--- graph2
|       ...
+--- L<dbi-connects|configfile-dbi-connects>
|       +--- connect1
|       +--- connect2
|       ...
+--- L<dbi-selects|configfile-dbi-selects>
|       +--- select1
|       +--- select2
|       ...
+--- L<discovery|configfile-discovery>
+--- L<environment|configfile-environment>
+--- L<general|configfile-general>
+--- L<groups|configfile-groups>
+--- L<host-templates|configfile-host-templates>
|       +--- host-template1
|       +--- host-template2
|       ...
+--- L<hosts|configfile-hosts>
|       +--- host1
|       +--- host2
|       ...
+--- L<html|configfile-html>
+--- L<links|configfile-links>
+--- L<ntops|configfile-ntops>
+--- L<oids|configfile-oids>
+--- L<page-templates|configfile-page-templates>
|       +--- page-template1
|       +--- page-template2
|       ...
+--- L<pages|configfile-pages>
+--- L<remotepings|configfile-remotepings>
+--- L<rrds|configfile-rrds>
|       +--- rrd1
|       +--- rrd2
|       ...
+--- L<run|configfile-run>
        +--- L<run-stages|configfile-run-stages>
        |       +--- stage1
        |       +--- stage2
        |       ...
+--- L<scripts|configfile-scripts>
|       +--- script1
|       +--- script2
|       ...
+--- L<times|configfile-times>
+--- L<tools|configfile-tools>
+--- L<view-templates|configfile-view-templates>
|       +--- template1
|       +--- template2
```

```
|       ...
+--- L<views|configfile-views>
|       +--- view1
|       +--- view1
|       ...
```

(You can look at the base configuration directory if you want, but you should also read through this so you know the significance of what you see.
Almost all the configuration files allow both blank lines and comment-lines. A comment-line **begins** with a # and the whole line is ignored by remstats. Inline comments are **not** permitted as the '#' is used in some places for other purposes. The only files which don't permit comments are the `view-templates`, which are html.
`alert-templates`, `customgraphs`, `dbi-connects`, `dbi-selects`, `hosts`, `host-templates`, `rrds`, `run-stages`, `scripts`, `views` and `view-templates` are sub-directories with the files within describing one of that kind of entity. E.G. a file in the `hosts` sub-directory is named for the host and contains that host's configuration.
NOTE: within the sub-directories, files with names beginning with 'IGNORE-' or ending with '~', will be ignored.
There are also a few *tools|config-tools* to help you make and update your config-dir, although not all parts of it.

Configuration - access

The first matching record, (all fields match or are "*", which matches anything), is used and grants the access specified.
Any of the fields may be "*" except the "access" field. Any fields which are not set to "*", must match the correcponding data. E.G. if the "program" is set to "xyzzy", then the record only applies to the "xyzzy" program and if all the other fields match, then the user is granted the access specified.
The fields are: access program user group ip/host.

"access" is either "allow" or "deny", with obvious meaning.

"program" is the name of the program (whatever is set in $main::prog).

"user" is from the CGI variable REMOTE_USER

"group" is a group that the "user" belongs to, as implemented in the htpasswd program supplied by the apache web-server.

"ip/host" is either a domain-name, a domain-name suffix (indicated by a leading "."), an IP-number, or an IP-number prefix (indicated by a trailing "."). A domain-name suffix matches if the name that REMOTE_ADDR resolves to ends with the suffix. Similarly, an IP-number prefix matches if REMOTE_ADDR begins with the prefix.

Configuration - Alerts

The alerts config-file is used by the *alert-monitor* to decide who to send alerts for problems. The *rrds|configfile-rrds* and *hosts|configfile-hosts* config-files decide `when` an alert needs to be raised, and these lines tell `who` gets the alert.
Each line is in seven parts, most of which are patterns, e.g.:

```
warn     * MISC UPTIME 0 0 uptime-alerts
error    silverlock.dgim.crc.ca  * * 600 900 test-alerts
error    news.crc.ca * * 600 900 news-alerts
critical * * * 600 900 critical-alerts
```

The first "word" is the status, as decided by the *alert-monitor*. The second word is a regex to match against the hostname. The third is a regex to match against the rrdname. The fourth is a regex for the variablename. The fifth is the minimum time for the alert condition to be present before an alert can be triggered. The sixth is the interval after sending an alert before another will be sent. The seventh is an `alert-destination` as specified in the *alert-destination-map|configfile-alert-destination-map*.
Note: The seventh used to be an alert program and there was an eighth which was an address, of a form appropriate to the alert program. This has been rolled into the `alert-destination-map` to make it more flexible.

If the current condition matches the status, host, rrd, and variable, then alert-monitor has to look at the times. If this is a new condition (i.e. it was in OK status previously), then an alert won't be triggered until after the minimum time has passed. This avoids transient problems being reported. If you want these to be reported, then set it to zero. If this is an old alert, then an alert won't be triggered until the interval time has passed since the previous alert. If the interval is 0 (zero) then there will only be one alert at the start-time.

Configuration - alert-destination-map

This config-file specifies the mapping from an abstract alert destination, specified in the *alerts config-file|configfile-alerts*, and the address(es) to send it to. The *alerter* attempts to match the abstract alert destination against each of the `map` lines and sends alerts to any which match.
There are three kinds of lines in this config-file: `map`, `alias`, and `method`. `Map` lines map from an abstract alert destination, listed in the alerts config-file, to a less abstract alias, listed here. The `alias` lines allow a crude list capability and also permit the use of different methods to deliver the alert. `Method` lines tell what program to run with what arguments in order to deliver to that type of address.

Map Lines

A map line looks like:

```
map DEST TIME DOW DOM MON ALIAS
```

Where:

DEST is an abstract alert destination, listed in the alerts config-file

TIME is a time-of-day specification, comma-separated time-ranges or '*' meaning all times. A range looks like HHMM-HHMM.

DOW is a day-of-the-week spec, a comma-separated list of weekdays, in numeric form (0=sunday, 1=monday, ...) or '*' for all weekdays.

DOM is a day-of-the-month spec. It's a comma-separated list of day-ranges, where a range is a day or DD-DD, or '*' ofr all days.

MON is a month spec. It's a comma-separated list of month-ranges, in numeric form, like MM or MM-MM or '*' for all months.

ALIAS is the alias that this DESTination maps to during the specified time-period. It's defined in an alias line.
This permits different DESTinations to be sent to different people at different times, depending on who's on duty.

Alias Lines

An alias line looks like:

```
alias ALIAS METHOD:ADDR ...
```

Where:

ALIAS is the alias being defined

METHOD is an alert-delivery method (see methods below)

ADDR is an address which is valid for that method
This indirection permits delivery of the same alert via multiple methods, in case one or more of the methods isn't available, as well as to different people.

Method Lines

A method line looks like:

```
method METHOD COMMAND-LINE
```

Where:

METHOD is the method being defined

COMMAND-LINE is the program to run with any arguments it requires. It will be passed the alert message on stdin and the address to send it to at the end of the COMMAND-LINE.

An Example

We have three guys on different shifts who manage network operations (Tom, Dick and Harry) during the week. On the week-end Frank is on call. We also want to email a copy to an email address which collects all the alerts. We want to send the alerts to whoever is working at the time. Say the abstract destination specified in the *alerts config-file|configfile-alerts* is `alerts`. We might use lines like those below:

```
map alerts 0600-1359 1,2,3,4,5 * * tom
map alerts 1400-2159 1,2,3,4,5 * * dick
map alerts 0000-0559,2200-2359 1,2,3,4,5 * * harry
map alerts * 0,6 * * frank

alias tom email:tom@our.com email:alert-history@our.com winpopup:console
alias dick email:dick@our.com email:alert-history@our.com winpopup:console
alias harry email:harry@our.com email:alert-history@our.com winpopup:conso
alias frank page:555-1234 email:alert-history@our.com winpopup:console

method email /home/remstats/bin/alert-email
method winpopup /home/remstats/bin/alert-winpopup
method page /home/remstats/binalert-page
```

Note: the hypothetical `page` method isn't provided with remstats. There are lots of different programs to send pages. Look at *alerter* if you want to add your own methods; it's easy.

Configuration - alert-template-map

The `alert-template-map` tells which *alert-template|configfile-alert-templates* to use for which addressee or RRD. The lines look like:

```
address regex template
```

or

```
rrd rrd template
```

or

```
rrd rrd:variable template
```

Addresses are checked first. This is to allow special mapping for devices like pagers which can't display a lot of information. If none of the special addresses match, then RRDs are checked, first with variables then without. An RRD can be an RRD instance, like `port-ftp`, or the wild RRD, e.g. `port-*`.
The `template` is the name of the template file in the `alert-templates` config-dir.

Configuration - alert-templates

The `alert-templates` directory contains the alert message templates. They are just text files with *magic cookies|cookies* in them. The cookies available are slightly different than the standard list, but they work the same way. You put ##COOKIENAME## wherever you want to see the value of the 'cookiename' variable. The ones available for alerts are:

HOST - the host for the alert

REALRRD - the RRD instance for the alert

FIXEDRRD - the RRD instance with the character-set translated a bit for file-names and message-id's

VAR - the variable name

STATUS - the alert status (OK, WARN, ERROR, CRITICAL)

VALUE - the value of the variable that caused this alert

RELATION and THRESHOLD - the alert is triggered when the VALUE is no longer in RELATION to the THRESHOLD value.

START - when the alert was first noticed

DURATION - how long the alert has been in this STATUS

HOSTDESC - the description line for this host

RRDDESC - the description for this instance of the RRD

NOW - the current time as a unix timestamp

NOWTEXT - the current time for email headers

ALERTHOST - the hostname of the host sending the alert

TOWHO - the addressee for this alert

There are three special files in the `alert-templates` directory, which **must** exist:

DEFAULT - which contains the default template to be used when no other matches the *alert-template-map|configfile-alert-template-map*.

HEADERS - which supplies the headers for each message, with the same substitutions as the rest of the template files. Make very sure that the HEADERS file ends with or contains an empty line or your message will be interpreted as part of the headers and will undoubtedly look wrong. The *alert-email* script does not check this.

FOOTER - supplies a standard ending for each message.

Configuration - Archives

The archives file names various data-retention periods.

RRDtool|http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/ calls them RRAs. Each line is in two pieces: an archive name and an RRA specification, exactly as documented in the rrdcreate manpage. Unfortunately, modifying this after the rrd has been created isn't one of the things that RRDtool does. I've got an rrd munger on my to-do list|todo, but it's still not done.

For example:

```
day-avg         AVERAGE:0.1:1:600
week-avg        AVERAGE:0.1:7:300
month-avg       AVERAGE:0.1:30:300
3month-avg      AVERAGE:0.1:90:300
year-avg        AVERAGE:0.1:365:300

day-min         MIN:0.1:1:600
week-min        MIN:0.1:7:300
month-min       MIN:0.1:30:300
3month-min      MIN:0.1:90:300
year-min        MIN:0.1:365:300
```

Configuration - Availability

There are two types of availability definitions: for an RRD or for an RRD on a particular host. The RRD may also be a wildcard RRD, like "df-*" or an instance of an RRD, like "df-/home". The definitions look like:

rrd RRDNAME VARNAME CF RELATION THRESHOLD and

host HOSTNAME RRDNAME VARNAME CF RELATION THRESHOLD

The CF is one of LAST, MIN, MAX or AVERAGE, with rrdtool's usual meaning. The RELATION can be any one of: <, <=, >, =, or =. The THRESHOLD is the number to which the value of VARNAME must be in the correct RELATION. (Clear as mud.)

As an example, take the following definition:

```
rrd ping rcvd MINIMUM > 0
```

This means that the variable 'rcvd' in the ping rrd must be greater than zero for it to be considered "available". All time intervals where it isn't, or for which no data is available, are considered "unavailable".

There are also two other record types: colors and thresholds. A colors record looks like:

```
colors COLOR1 ...
```

A thresholds line looks like:

```
thresholds NUMBER ...
```

and must have the same number of values as the colors line. Only one of each. Here's an example to make the use clear (I hope):

```
colors avail1 avail2 avail3 avail4
thresholds 99 98 95 90
```

The `colors` line above requires that the colors ‘avail1‘, ... be defined in the *colors|configfile-colors* config-file. The `thresholds` line above specifies that if an availability is 99% or above, it should be colored ‘avail1’ color, 98% to 99%, use ‘avail2’ color, etc.

Configuration - Colors

The `colors` config-file just gives names to colors so you don‘t have to remember or decipher hex numbers. Each line is: a colour name and a six-digit hex number giving the RGB values. The color "down" is special and is used in the ping index to color the background of hosts which are down.
For example:

```
totalcolor      00a000
usedcolor       a00000
downcolor       a07777
```

Configuration - Customgraphs

The `customgraphs` directory contains files which define graphs which aren‘t associated with a specific host. They are very like a graph defined on an *rrd|configfile-rrd*. They do have a times line preceeding the graph definition, to set the time-periods for that graph. They are linked in under the `Custom Index`.
Note that the graph definition itself must be indented, like rrd graphs.
As an example:

```
times   day yesterday week 3month year
        --upper-limit 100 --lower-limit 0 --rigid
        --vertical-label 'CPU %'
        --title 'CPU Usage (##GRAPHTIME##)'
        DEF:silverlockuser=##DATADIR##/silverlock.dgim.crc.ca/cpu.rrd:user:AVERAGE
        DEF:silverlocksys=##DATADIR##/silverlock.dgim.crc.ca/cpu.rrd:system:AVERAG
        DEF:loisuser=##DATADIR##/lois.dgim.crc.ca/cpu.rrd:user:AVERAGE
        DEF:loissys=##DATADIR##/lois.dgim.crc.ca/cpu.rrd:system:AVERAGE
        CDEF:silverlock=silverlockuser,silverlocksys,+
        CDEF:lois=loisuser,loissys,+
        'LINE2:silverlock###COLOR1##:silverlock'
        'LINE2:lois###COLOR2##:lois'
```

dbi-connects

The `dbi-connects` config-dir is used by the *dbi-collector* to specify how to connect to a database. Each file in this directory is named for the connection, which is used in the RRD definition. There are three directives which may occur in these files:

dbisource - specifies a DBI source string. There are three parts to it, separated by colons (‘:’). The first part is always ‘`dbi`’. The second part is the DBD (DataBase Driver) name, e.g. ‘Oracle’ for an Oracle database. The final part is database-specific, so you‘ll have to look it up under `DBD::Whatever`. As an example, for Oracle, you might have a `dbisource` like:

```
dbisource dbi:Oracle:MyDbName
```

or

```
dbisource dbi:Oracle:host=my.host.com;sid=MYSID
```

user - specifies the username to use to connect to this database

password - specifies the password for that user

The `user` and `password` are optional, and can be specified on the RRD directive of the appropriate host‘s config-file. In addition, the string can be parameterized using remstats usual *magic cookies|cookies* mechanism, in this case restricted to USER, PASSWORD, SELECT, DATABASE and HOST.

dbi-select

The `dbi-select` config-dir contains files which each contain a single SQL select statement, formatted however you like. An rrd which is collected by the `dbi-collector` will specify both a `connect` directive (*dbi-connects config-dir|configfile-dbi-connects), and a `select` directive. The `select` directive specifies the name of the select, which is defined in a file of the same name. All names are lower-cased, so do not make file-names of mixed or upper case.*

environment config-file

Description:

It contains environment variables and values to be supplied by *run-remstats* to its children. Each line looks like a name, whitespace and everything else on the line is the value:

```
XYZZY   Everything else on the line
```

Configuration - General

This is the miscellaneous config-file, but there are some critical pieces here:

`datadir` (REQUIRED) - The data for a given host is stored under `datadir/hostname`. There are also other status files stored in this directory.

`staletime` (UNUSED) - How long before we count a status as stale. (seconds)

`minuptime` - How long a host must be up before it stops being flagged as recently up. (seconds)

`keepalerts` (UNUSED) - How long to keep records of alerts after the condition no longer exists. (seconds)

`uptimealert` - If this is set, the *alert-monitor* will cause a warning level condition on the fake rrd MISC for the fake variable UPTIME, for any host whose uptime is less than this value. Whether this will trigger an alert depends on the *alerts|configfile-alerts* file.

`pinger` - If defined, this names the *ping-collector* to be used before all the other *collectors*. (Unless you write your own, you put `ping-collector` here.) If you don't include this line, then you'll want to make sure the the `ping-collector` is listed in the `collectors` line (below).

`collectors` - This line tells *run-remstats* which collectors to run. The default list is all of them, so you can gain some benefit by pareing this line down to those you are using, but remember it if you add new *rrds|configfile-rrds* that need other collectors later. **Note:** you list the names of the collectors without the '`-collector`' on the end. E.G. the `ping-collector` would be included as just '`ping`'.

`monitors` - This line tells *run-remstats* which *monitors* to run. The default is all of them.

`pagemakers` - This tells *run-remstats* which *pagemakers* to run at the end, if the config-dir has changed. The default is all of them.

`max-port-patterns` - This tells the *port-collector* how many parenthesised patterns there can be, at most, in `valuepatterns` or `infopatterns`. The default is 10.

`watchdogtimer` - This sets the limit that *run-remstats* will apply to each of the programs that it runs, so that, e.g., a hanging collector will not hang the whole remstats cycle.

`keeplogs` - This tells how long *cleanup* will permit old files to hang around, in seconds.

`pagetemplatedir - This tells where the page templates for page-writer are kept. This allows you to change the look of pages with a one-line change here. Assuming that you have a different set of page templates in another directory.`

Configuration - Groups

If any name has spaces, it must be 'quoted' or "quoted". Any groups not listed here U<will not> be linked into the HTML index pages generated by *page-writer*, but data will still be collected. If there is a file in the `htmldir` with the same name as a group-name, but with the spaces replaced by '_' and all the letters lower-cased, followed by '.html', then the group-names in html pages will be linked to those pages. E.G. for a group named `"Local Routers"`, if there is a file called `local_routers.html`, the name of the group will be made into a link to that file.

Configuration - Hosts

The `hosts` files are what the whole configuration has been working toward. Here we tell which hosts we're interested in and what we want to monitor. Here's a sample host file called `clark.dgim.crc.ca`:

```
        desc    DNS and Web
        ip      142.92.39.18
        aliases ns1.crc.ca
```

```
via     142.92.32.10
keys    server sun
group   Servers
contact Thomas Erskine <thomas.erskine@crc.ca>
tools   ping traceroute telnet http clark-special:special
rrd     ping
rrd     cpu
noalert cpu user
community       xyzzy
rrd     load
nograph load users
rrd     if-le0
alert   if-le0 ierr < 1000 5000 10000
alert   if-le0 in WARN
rrd     df-/var
rrd     df-/tmp
rrd     port-http critical
rrd     port-ssh
rrd     port-whois
noavailability port-whois status
noavailability port-whois response
rrd     port-domain critical
```

The name of the file (`clark.dgim.crc.ca`]) is the host that you're interested in. The name should be a fully-qualified-domain-name, but anything which perl's getaddrbyname can resolve should work.
The `ip` line saves the IP number from having to be looked up and could be used to deal with hosts which aren't in the DNS. If you want the IP number to be looked up each time, you can leave this line out.
The `desc` line gives this host a description *page-writer* will put on pages about this host.
The `alias` line tells remstats about other names for this host. This is mainly for the `ping-collector` to allow it to tell for sure when it has got a response from this host.
The `via` line is used by the *topology-monitor* to specify networking gear (like hubs and switches) which are in the path to the host, but won't show up in a traceroute.
The `keys` line is used to attach arbitrary attributes to hosts, for selection purposes. This is to make it easier to write scripts which act on specified collections of hosts. For an example, see the *update-switch-ports* script.
The `group` line is required and tells which group this host belongs to. Remember, you defined all the groups back in the *groups|configfile-groups* file?
The `contact` line tells who to contact for this host. If a line in the *alerts config-file|configfile-alerts* refers to a recipient called `CONTACT`, the value of the host's contact line will be substituted.
The `tools` line tells which tools (defined in the *tools config-file|configfile-tools*) you want to appear for this host. E.G. if a host doesn't have a web-server, there's no point in providing a link to connect to it. To accomodate host-specific tools, a toolname can be given as `real-tool-name:display-name`. This means that the tool will be defined in the `tools` config-file as `real-tool-name`, but will be displayed as `display-name`.
The *rrd|configfile-rrds* lines tell which rrds to collect for this host. If the rrd was defined as a wildcard, it will have the instance specified here. In the example there are three wildcard lines, referring to `if-le0`, `df-/var` and `df-/mail`. The first is looking at the data for network interface hme0 and the others are getting data on the /var and /mail file-systems, respectively.
[Alert definitions are fully specified in *alert-monitor*.]
The first `alert` line is setting the alert threshold for `if-le0` to 50. If this host file was from the same configuration as the previous `rrd` sample, the alert here would override the one in the `rrd` file. There is also a `noalert` line, which cancels an alert set in the rrd without setting a replacement alert. The alert line for a host must specify the rrd as well, but is otherwise the same as an alert on an rrd.
The second `alert` line is specifying the status (`WARN`) for missing data for the `in` variable.
There can also be descriptions for rrds. If you append to an `rrd` line something like `desc='xyzzy'`, then you'll see that description on pages dealing with it. I added this for labelling network interfaces, but you can use if for anything you want.

The `community` specifies the SNMP community string to use for this host to fectch SNMP data. If the host config-file doesn‘t specify any RRDs collected by the snmp-collector, you don‘t need to specify a community.
If this host uses any rrds collected by the snmp-collector, it can also specify a port to use like:

```
snmpport        3401
```

If the RRD itself specifies a port, then the RRD-specified port will be used instead, for that RRD.
If you don‘t want a particular graph for this host, you can include a `nograph` line. It looks like:

```
nograph rrdname graphname
```

There can also be a `statusfile` line, looking like:

```
statusfile NNN
```

with `NNN` replaced by the name of a status file from that hosts‘s data directory. This permits the main index pages to show the status of an un-pingable host as the status of something else, like the reachability of it‘s web-server (STATUS-port-http).
Related to that, there can be multiple `headerfile` directives, which look like:

```
headerfile FFF LLL
```

with `FFF` being the name of a file in that host‘s data directory, and `LLL` being a label for the header. This can be used to inject arbitrary information into the header.
The `noavailability` lines tell the *availability-report* program not to report on certain rrd/variable combinations. In this case, we don‘t want to see availability stats on the whois server. Maybe it‘s too embarassing?

RRD-specific Information

There may also be RRD-specific information on an `rrd` directive, anything after the rrd-name.
Any of them can include "`desc="whatever"`" as part of the extra information. This description will be included in the headers of graph pages for that RRD.
For RRDs collected by the *port-collector*, there may optionally be the string `critical`, which will cause a status of ERROR to be elevated to CRITICAL. This is a relic, which may go away, so don‘t count on it. Instead, use the `alert` directive to override the RRDs alert specifications.
For RRDs collected by the *log-collector*, the extra information is not optional, and consists of the full path to the log-file from which that RRd‘s information will be collected.
For RRDs collected by the *dbi-collector, there may be several different pieces of extra information, like:*

```
rrd rrdname CONNECT="cc" SELECT="ss" USER="uu" PASSWORD="pp" DATABASE="dd"
```

This permits using an RRD definition while overriding many parts of that definition.

Configuration - Host Templates

These config-files simply hold the same kind of lines as a *host config-file|configfile-hosts*. By adding a line like:

```
template some-host-template
```

to a host config-file, you achieve the same effect as adding all the lines contained within the template file. If you have many hosts which are similar, this can be a usefull way of keeping the configuration consistent.
It can also be used to parameterize things. As an example, if you are using the *nt-status-server* and are only running it on a single host which is providing information on various other NT hosts, you might make a template, say `default-nt-status-server` like:

```
nt-status-server my.nt.status.server
```

and replace the `nt-status-server` lines in those hosts with:

```
template default-nt-status-server
```

Then if you want to change which machine is running the `nt-status-server`, you‘d just have to change the template.

Configuration - HTML

The `html` file defines stuff related to web-page generation. There are several different kinds of information.

Locations

These things define where things are, like URLs. They are:

`htmldir` (REQUIRED) - The html stuff for a given host is stored under `htmldir/hostname`.

`htmlurl` (REQUIRED) - How to refer to the `htmldir` in a URL.

`cgidir` (REQUIRED) - The static cgi scripts, like *ping.cgi|ping-cgi*, live here.

`cgiurl` (REQUIRED) - How to refer to the `cgidir` in a URL.

`viewdir` - Where to store the views, in case you don't want them under the `htmldir`.

`viewurl` - How to refer to the `viewdir` in a URL.

`webmaster` (REQUIRED) - Who's in charge of these web-pages, an email address to get stuffed into mailto URLs.

`logourl` - Where is the logo for this site

`homeurl` - ehere is home for this site

`topurl` - where top goes for this site

`rrdcgi` - where to find the rrdcgi program, I like to link it and rrdtool into `/usr/local/bin`, for ease of use.

`motdfile` - where to find the Message-Of-The-Day file. This is used to add in announcements at the top of the index pages, except the host index.

"Look and Feel"

`thumbnail` - How big the graph portion of a thumbnail image is to be (WIDTHxHEIGHT)

`metadata` - Where to store CERN-style meta-data, to set expiry times for the gifs. (METADIR METASUFFIX)

`background` - what should the background look like. It's mostly obsolete, because you can get the most of the same effects by editing the `default.css` style file instead.

`htmlrefresh` - How often to cause the web pages to refresh themselves. (seconds)

`upstatus`, `upunstablestatus`, `downunstablestatus`, `downstatus`, `okstatus`, `warnstatus`, `errorstatus`, `criticalstatus` - HTML to display for various statuses. The defaults use tags.

`viewindices` - Should *view-writer* write the index links at the top of view pages? (yes or no)

`showinterfaces` - Should *graph-writer* show interfaces on a host page? (yes or no)

`keepimages` - How long *cleanup* will permit old images to hang around, in seconds.

`default-tools` - What tools to show for a host which doesn't specify any.

Markers

This group supplies html to wrap various things in the generated web-pages.

indexprefix, indexsuffix - for the items on the `Indices` line of the header

groupprefix, groupsuffix - for the group names on the various indices

hostprefix, hostsuffix - for the host names on the various indices

toolprefix, toolsuffix - for the tool names on the toolbar

linkprefix, linksuffix - for the links in the footer

outofrangeprefix, outofrangesuffix - for the current value on the availability pages when it has gone outside the specified bounds. (See *availability-report*.)

Labels

If you translate the labels, the web-pages should be translated. It doesn't include error-messages or debugging messages.

The currently available ones, with their defaults, are:

alertreport	Alert Report
comment	Comment
contact	Contact
customindex	Custom Index
description	Description
error	Error
groupindex	Group Index
hardware	Hardware
hostindex	Host Index
indices	Indices
ipnumber	IP #
lastupdateon	This page last updated on
links	Links
logreport	Log Report
operatingsystem	Operating System
overallindex	Overall Index
pingindex	Ping Index
quickindex	Quick Index
status	Status
tools	Tools
uptime	Uptime
viewindex	View Index

And also the:

`uptimeflag` - shows on some index pages when a host has been up for less than `mintime` (defined in the *general|configfile-general* file.)

`alertflagwarn`, `alertflagerror` and `alertflagcritical` - give HTML to be inserted in the quick index for hosts which have alerts active.

Configuration - Links

The links config-file supplies links that you want to put with the standard links at the bottom of the web pages. They're in two or three pieces. The simple (two-part) form is text and a URL to link to, like:

```
SourceWorks http://www.sourceworks.com/
```

The third part, if present, is assumed to be a URL for an image if it begins with "http:", "ftp:", '/', or '../'. Otherwise it is taken as literal HTML and inserted without addition.
So, a simple link definition, like:

```
name url
```

produces HTML like:

```
<A HREF="url">name</A>
```

A three-part link with an image, like:

```
name url imageurl
```

produces HTML like:

```
<A HREF="url"><IMG SRC="imageurl" ALT="name"></A>
```

Otherwise, a link definition like:

```
name url html
```

produces:

```
<A HREF="url">html</A>
```

ntops config-file

This simple config-file tells the *ntop-collector* where the hosts running an ntop server are. Each line of the file contains a hostname and the port number on that host where the ntop server is running.

Configuration - OIDs

[These are for SNMP and you can ignore this config-file if you're not interested.]
The SNMP implementation in the snmp-collector is primitive and only knows about OIDs (Object IDs) by their number. Since I'm not interested in bringing in a full MIB compiler to deal with the MIBs, this section lets you specify names for the OID numbers you're interested in using later. The lines look like:

```
CiscoCpuLoad    1.3.6.1.4.1.9.2.1.58.0
```

for a non-hypothetical example, if you happen to have Cisco routers. If you have the ucd-snmp package, their snmptranslate program comes in handy for pulling out the appropriate numbers without the bother of tracking through the wretched MIBs.

the pages config-file

This file specifies the pages to be generated by *page-writer* and requires knowledge of it. It is processed sequentially from the beginning. Each line has one directive. The directives are taken from the tags processed by `page-writer`:

var VARNAME VALUE - this creates a variable called `VARNAME` and gives it the value `VALUE`

var::config VARNAME CONFIGVALUE ... - this creates a variable from the values in the configuration hash. In theory, any value can be obtained, but in practice two levels seems to be sufficient.

page TEMPLATENAME FILENAME - This causes the generation of a web-page using the `TEMPLATENAME` template from the *page-templates config-dir|configfile-page-templates* which will be stored in the file `FILENAME`

chdir DIRNAME - Changes to the directory `DIRNAME`, creating it if necessary.

Blank lines and lines beginning with a # are treated as comments and ignored.

The page-templates config-dir

This directory contains the templates referred to in the *pages config-file|configfile-pages* and used by the *page-writer*. You need to read the documentation for all three of these together.
Each file contains a single template. The template is HTML with the addition of some special <REMSTATS::xxx> tags. The tags are documented in the docs for `page-writer` as it implements them.

Configuration - remotepings

The `remotepings` file simply lists all the machines which are running the remoteping-server. Or at least all the machines that you want to query.

Configuration - Run

This config-file lists the *run-stages|configfile-run-stages* which *run-remstats2* will run, giving the order to run them in. It also specifies special conditions required to run them. Each line looks like:

NAME - the name of the stage to be run

WHEN - the names of conditions required to run this stage. Currently, there is only one: `CONFIGCHANGE`

Configuration - Run-Stages

This sub-directory contains files, each defining the programs to be run for a given run stage. [The order of the run stages is listed in the *run config-file|configfile-run*.] Each line contains to following fields, whitespace separated:

NAME - This gives this line a name, which will be used for error-messages and file-names. It's used instead of the program-name to enable multiple instances of a given program, primarily the collectors.

ASYNC - If this flag is set (1, y, yes or true), then *run-remstats2* will not wait for it to complete before moving on the the next stage. This permits long-running programs, on which nothing depends, to run through several stages.

FREQUENCY - This is the minimum interval between runs. `run-remstats2` will not run this

line if it has been run within this time.

COMMAND - This is the command-line to run. It may be a pipeline. The command will be augmented by the addition of output redirection for stdout and stderr to `@@TMPDIR@@/NAME.PID`, where PID is the process-id of the instance of `run-remstats2` which is running it.

Notes:

Any command can have the following `##MAGICCOOKIES##`:

RUNPID - is the process-id of the *run-remstats2* process which is invoking the program.

Configuration - RRDs

[Note: there is a *list of supplied rrds|rrds* with descriptions.]

These files are the most complicated. Here's an example, again taken from the `if-` rrd supplied with remstats.

```
collector       unix-status
step            300
keys                    xyzzy
data            in=interface_packets_in:* COUNTER:600:0:U
data            ierr=interface_errors_in:* COUNTER:600:0:U
data            out=interface_packets_out:* COUNTER:600:0:U
data            oerr=interface_errors_out:* COUNTER:600:0:U
data            coll=interface_collisions:* COUNTER:600:0:U
alert           in < 100
alert           out < 100
alert           in nodata WARN
archives        day-avg week-avg month-avg 3month-avg year-avg
times           day yesterday week month 3month year
graph           if-* desc='Interface data for ##RRD##'
                --title 'Interface ##RRD## ##GRAPHTIME##'
                --lower-limit 0
                --vertical-label 'packets'
                DEF:in=##DB##:in:AVERAGE
                DEF:out=##DB##:out:AVERAGE
                DEF:ierr=##DB##:ierr:AVERAGE
                DEF:oerr=##DB##:oerr:AVERAGE
                'LINE1:in###COLOR1##:Input Packets'
                'LINE1:out###COLOR2##:Output Packets'
                'LINE1:ierr###COLOR3##:Input Errors'
                'LINE1:oerr###COLOR4##:Output Errors'
```

This example shows most things that can be done, except multiple graphs on the same rrd, which is as simple as adding another graph line and its definition.

First, the rrd name is special, in this case. Any rrd definition file which ends in a '-' is assumed to be for a wildcard rrd, in this case `if-*`. This avoids problems with file-systems which are overly fussy about which characters can be in file-names.

This rrd definition will match any rrd beginning with 'if-' specified in a host config-file. Wildcard rrds are necessary when a given host may have more than one of whatever the rrd is referring to, in this case network interfaces. The network interface name will replace the '*' in the rrd line in the host config-file. It will also be available in the `##WILDPART##` *magic cookie|cookies*.

The `collector XXX` tells which *collector|collectors* supplies the data for this RRD. `collector unix-status` means that this RRD gets its data from the *unix-status-collector*. [This used to be called `source`.]

The `step` line sets the step value for the rrd. This is the expected frequency of data updates. [See the manpage for rrdcreate.] N.B. Setting this is required, but changing some RRDs won't change how often the collectors run. You'll have to change the crontab entries which run *run-remstats2* to specify the shortest interval. However, in the *run-stages config-files|configfile-run-stages* you can specify how often certain lines should be run.

The `keys` line attaches an arbitrary attribute to this rrd. This can be used by various programs to select subsets to operate on. E.G. *page-writer* can be told to use this to select only some RRDs.
(FIXME - the writing stinks)
The `data` lines define various DS elements for this RRD. [See the manpage for rrdcreate.] The first part is the DS name, with an extension. The collectors produce long names and may have instance-names added to the variable name, in this case to tell which interface this data is for. So the first part looks like `dsname=variable:instance`. The `dsname` is used for the RRD DS name and the `variable:instance` part is used to tell updater which collector information applies to this DS. Most of the rest of the line is straight from rrdcreate's description of DS.
It's also possible to invoke remstats *functions* or configuration-supplied *private functions|private* on the incoming raw data. The `data` line would look like:

```
data    xyzzy=&function(variablename) ...
```

It's your responsibility to make sure that `function` is available and that it returns a valid number.
There is a pseudo-function builtin to the *updater* called `FAKE`. Its purpose is to allow testing of alerts, but it could also be used to allow some foreign system to inject data, by putting its values there. It allows you to replace collected values with values of your choice.
It will look for a file in the host's data directory called `FAKE-rrdname-varname` with `rrdname` and `varname` munged in the usual fashion to make an acceptable file-name. If the file is there, then its content will be returned (with leading and trailing whitespace removed) as the value. If the file is not there, then the collected value will be returned.
The `alert` lines are setting the thresholds for alerts, in this case for the variables `in` and `out`. They must specify, in order: the variable-name, the relation (<, =, >, delta< and delta>) and a space-separated list of thresholds. Since these examples only supply one number each, they will only set OK or WARN statuses. If the variables `in` or `out` have values less than (<) 100, they are considered to be OK. Otherwise they're elevated to WARN status. What will happen when they go into WARN status depends on the *alerts|configfile-alerts* file. These alerts will apply to any host which uses this rrd, unless the host overrides it.
The last alert specifies that missing data for the variable `in` will be considered to be status `WARN`, for purposes of generating alerts. The full description of the alerts is kept in the docs for *alert-monitor* as it is the program which implements them.
The `archives` line tells how to keep the data for this rrd, using the names defined in the *archives|configfile-archives* file.
There can be multiple `graph` lines describing as many graphs from the data in this rrd as you want. The graph-name must be wildcarded if the rrd is. A `graph` line is followed by its definition which must be indented. The definition is straight from rrdgraph with the *magic cookie|cookies.html* substitution. If you want a description , you can add:

```
desc='whatever you want'
```

or

```
desc="whatever you want"
```

to the `graph` line. This is used to set the alt text on the web-page.

The `currentvalue` Directive

It looks like:

```
currentvalue  STATUSFILENAME VARIABLENAME
```

or

```
currentvalue  STATUSFILENAME &FUNCTION(VARIABLENAME)
```

This permits the RRD definer to have the current value of an RRD variable stored externally in a status file in the host's data directory. As an example, the supplied `apcupsreplacebattery` RRD definition arranges to store the current value of the indicator in a status file. A host using this RRD can then use the `headerfile` directive to arrange for this indicator to be displayed in the "host" (in this case a UPS) headers.

Collector-specific Stuff

Any `data` line can have extra stuff added to the end of the line. What this extra stuff means depends on the RRD.

Any data collected by the *unix-status-collector* will have a `section-name` added to the end, to be passed on to the *unix-status-server* to tell it which sections need to be run for this host. Some sections will also specify extra information. E.G. the `procname` section will need the name of the process to be counted. [See the *unix-status-server* for complete specifications.]
An rrd collected by the *log-collector* will have extra stuff on each `data` line after the DS information. The extra stuff will be the function and pattern needed by log-collector to pass to the *log-server* to get that variable's data.
An rrd collected by the *port-collector* may specify that this particular service is critical, by simply including the word "critical" at the end of line. This will cause the status to be elevated to CRITICAL status if the status ever reaches ERROR level.
An RRD collected by the *snmp-collector* needs to specify which OIDs to fetch. They are specified by name in the RRD with a line like:

```
oid APCUpsAdvInputLineVoltage
```

which refers to a name defined earlier in the *oids config-file|configfile-oids*.
An RRD collected by the `snmp-collector` may also specify an SNMP port to use with a line like:

```
port    3401
```

For an RRD collected by the `dbi-collector`, the `data` directive will require the column number in the select that will provide its data. This will be appended to the `data` directive like:

```
data xxx=yyy GAUGE:600:0:U COL=12
```

There are also pseudo-columns called `STATUS` and `RESPONSE` to enable the rrd definition to reference the connection-status and response-time.

List of RRDs

Collected by the *cisco-access-collector*

ciscolinespeeds - distribution of line speeds used on a Cisco Access Server. It worked on a 5300 and ought to work on the 52xx series.

collector-cisco-access - status of the cisco-access-collector itself

Collected by the *log-collector*

collector-log - status of the log-collector itself

dnscachelog - counts of access to dnscache (from djbdns)

ftpxferlog - file and byte counts from xferlog produced by wu-ftpd

httpdlog - from a standard httpd log-file, counts accesse, successes and bytes

namedstats - stats from a BIND8 server, triggered by signalling it every 5 minutes. Only possible on a lightly used server.

newshistory - counts articles by hierarchy, from an INN server's history file. You'll have to change this to include hierarchies which are of interest to you.

newslog - from the newslog of INN, counts articles by news-neighbor. Again, you'll have to change this to include your neighbours.

qmaillog - from qmail-send's log, counts messages in, deliveries (ok, failed or deferred), and concurrency of local and remote deliveries.

qmail-poplog - from a tcpserver log of running qmail-pop3d, counts sessions and concurrency.

qmail-smtplog - from a tcpserver log of running qmail-smtpd, counts sessions and concurrency.

sendmaillog - from a sendmail log, counts messages, bytes and deliveries. Not tested recently; let me know if it still works, or not.

Collected by the *nt-status-collector*

collector-nt-status - shows the status of the nt-status-collector itself

ntactivity - from performance counters, gives interrupts, context-switches, system-calls, sessions and open-files counts.

ntcpu - from performance counters, gives cpu usage. Unfortunately, I have yet to figure out

what these numbers actually mean. Explanations and/or calculations are welcome.

ntlogicaldisk-* - from performance counters, gives reads, writes, bytes in and out and disk free.

ntmemory - from performance counters, gives memory free and committed.

ntnetwork - from performance counters, gives network bytes and packets in and out, for the host as a whole. Not very usefull for a host with more than one interface.

ntnetworkinterface-* - from performance counters, gives, for input and output, bytes, unicast packets, non-unicast packets, errors and discards.

ntpaging - from performance counters, gives page reads and writes, count of pages in and out and page fount counts.

ntservice-* - from SRVINFO (in the NT Resource Kit), gives status of the named NT service.

nttime - from time, gives time difference between NT host and remstats collector host.

ntuptime - from SRVINFO (in the NT Resource Kit), gives uptime

Collected by the *ping-collector*

collector-ping - the status of the ping-collector itself

ping - gives ping-echo's received and min, max and average of response time for them.

ping-* - as ping, but allows you to ping a specified interface on this host.

Collected by the *port-collector*

collector-port - gives the status of the port-collector itself.

airqualityontario-* - collects air quality info from the Ontario Ministry of the environment web-site, giving a number which indicates how bad the air is.

environmentcanada-* - collects weather info from the Environment Canada web-site, by airport code, and gives status and response-time, temperature, pressure, windspeed, windchill, dewpoint, humidity and visibility. An example of a wildcard port collector.

port-* - gives status and response-time for the specified port

weathernetwork - collects weather for Ottawa from The Weather Network's web-site and gives status and response-time, temperature, humidity, humidex, pressure, windspeed, and dewpoint.

Collected by the *snmp-collector*

apcenv - collects info from an APCC UPS's Environmental Monitoring unit, giving temperature and humidity.

apcups - collects ups-related info from an APCC UPS's SNMP unit giving time on battery, battery capacity, battery temperature, battery run-time remaining, load, output current, voltage and frequency and input voltage, current, frequency and phase.

bgppeer-* - collects info about BGP4 peers via SNMP giving current BGP state, counts of updates and sessions, peer start time and last update time.

ciscotemperatures - collects info from high-end Cisco routers giving the inlet and outlet temperatures for the chassis.

collector-snmp - shows the status of the *snmp-collector* itself.

dsinuse - collects the counts of lines in use from a Cisco 5300 Access Server via SNMP (ought to work on a 5200 too).

frif-* - collects info on Frame-Relay interfaces, similar to snmpif-*, giving input and output bytes and frames, speed and operational status.

netappcpu - collects CPU usage from a Network Appliance Filer.

pixmem - collects memory usage info from a Cisco Pix Firewall.

pixsessions - collects session counts from a Cisco Pix Firewall.

snmpcpu - collects CPU usage from Cisco Routers.

snmpif-* - collects interface data giving input and output bytes, unicast and non-unicast packets, errors and interface status.

snmpif2-* - collects interface data giving input and output bytes, unicast and non-unicast packets, errors and interface status. This rrd is intended to replace snmpif-* when it's working

correctly, as it doesn't require special-casing in the snmp-collector.

snmpmem - collects memory usage from Cisco routers

snmpuptime - collects uptime

squid - collects usage info from te squid web-cache giving memory, CPU and disk usage, number of objects in the cache, cache hits, bytes in and out and current number of clients.

Collected by the *snmp-route-collector*

collector-snmp-route - shows the status of the *snmp-route-collector* itself. Note that this collector is too slow to be of interest except for routers with a small number of routes, not Internet gateway routers.

snmp-routes-* - collects counts of routes and best routes from a given peer.

Collected by the *unix-status-collector*

afpdprocs - (obsolete) collects count of AppleTalk File Protocol Daemons running (from netatalk).

atalkdprocs - (obsolete) collects count of AppleTalk Daemons running (from netatalk).

collector-unix-status - shows the status of the *unix-status-collector* itself.

cpu - collects user and system cpu usage.

df-* - collects disk (file-system actually) size and free space in bytes and (if applicable) inode totals and free counts.

ftpcount - collects count of wu-ftpd daemons running, by access class. Note that you'll have to modify this one to specify the access classes of interest.

ftpusers - collects counts of wu-ftpd daemons running.

if-* - collects info from "netstat -i" interface packets and errors, both input and output, plus collisions and interface status.

inndsize - collects the size of the INN News Daemon

load - collects the load average for 1, 5 and 15 minutes.

memory - collects the memory usage from vmstat, including scanrate on Solaris.

namedxfers - collects the count of BIND's zone transfers running.

newsreaders - collects the count of news readers, actually the count of nnrpd processes.

nmbdprocs - (obsolete, replaced by procname-*) collects counts of nmbd processes running (part of the samba package).

proc-diskio - collects disk I/O info from /proc/stat on linux giving read and write operations and block counts.

procname-* - collects a count of the number of the * named process.

qmail-qstat - collects info about the qmail queue giving size and backlog.

qmail-qstat2 - collects info about the qmail queue giving size and backlog. This version operates without calling `qmail-stat`, to avoid a bug in it when dealing with large queues.

qmail-qtypes - (obsolete, use qmail-qstat)

sarlin-paging - SAR stats for Linux paging, showing both activity and occupancy type.

smbdprocs - (obsolete, use procname-*) collects a count of the number of smbd processes running.

tcpstates - collects counts of tcp sessions in their various states from netstat giving established, syn-received, listen, close-wait, last-ack, fin-wait-1, fin-wait-2, closing, time-wait, and syn-sent.

unixtime - collects the difference between the collector's clock and the collected system's clock.

uptime - collects uptime (number of seconds).

Configuration - Scripts

The `script XXX` files are describing how to query a given port for its status and are used by the *port-collector*. They look like:

```
send    GET / HTTP/1\.0\n\n
timeout 5
```

```
port    80
infopattern ^Server:\s+(.*)$
valuepattern ^Content-length:\s*(\d+)
ok      ^HTTP/\d\.\d 200
warn    ^HTTP/\d\.\d [45]\d\d
```

This example is taken from the supplied `config-base` and queries an HTTP server for its root page. First, it sends the "send" text, which in this case is a minimal HTTP request, and waits no more than 5 seconds. After the port is closed from the remote end, or the timeout expires, any text which was returned is examined by the various tests. In this case, if the web-server sends back a line beginning something like "HTTP/1.1 200", the port will be marked as "OK". Similarly, there are "warn", "error" and "critical" statuses possible.

The `port` is optional and `getservbyname` will be called on the script name, if port isn't specified. This also lets you have multiple scripts for the same port, using different names for the script.

The `infopattern` is optional, and supplies a pattern which will be matched against each line in the result. If there is a match, files will be created in the data directory for that host called `INFOn-rrdname`, where `n` will be in the range 1..9 and `rrdname` will be the name of this rrd, converted to a file-name. The files will contain matches for parenthesised items in the regular expression. E.G. in the example above, a file will be created called `INFO1-http` which will contain whatever the web-server said its type and version was.

Similarly, the `valuepattern` is also optional, but the matches will be returned as collected items called `value1` through `value9`. In the example, this would cause the collector to return a line like:

```
hostname timestamp value1 1022
```

An RRD definition could use this by including a line like:

```
data pagesize=value1 GAUGE:600:0:10000
```

For a working example, look at the RRD definition for weathernetwork.

Configuration - Times

The `times` file specifies time intervals for which graphs will be made. I suppose it should be renamed graphtimes or something, but I've got other things to do. Each line is in three pieces: a time name, a start time and and end time. The times are relative to the current time and so will always be non-positive.

The currently defined times are:

```
thumb           -60*60*2 0
day             -60*60*24 0
week            -60*60*24*7 0
month           -60*60*24*30 0
3month          -60*60*24*30*3 0
year            -60*60*24*365 0
yesterday       -60*60*24*2 -60*60*24
lastweek        -60*60*24*7*2 -60*60*24*7
```

Note:

The times `thumb` and `day` are special. The *graph-writer* expects them to exist and to have certain meanings. The `thumb` time is a short interval which is used to make the ping thumbnail graphs for the ping index. The `day` time is the default time interval. The higher-level pages will use the `day` graph as a link to the other time intervals.

Configuration - Tools

The `tools` file is only used by *graph-writer* to create toolbars. Each line is in two pieces: a tool-name and a URL to link to for this tool.

The URL can have *magic cookies|cookies* in it to substitute in things like hostname. Currently, the only cookies which will get substituted here are `HOST`, `IP` and `HTMLURL`. If you think of other usefull ones, please *tell me|mailto:terskine@users.sourceforge.net*.

Views - your own selection of graphs on one page

The views config-dir contains files, one per view, describing a collection of things that you want to see on one page. There are three kinds:

simple - you specify which graphs or customgraphs you want, using lines like:

```
graph        hostname rrdname graphname
customgraph  customgraphname
```

You can have as many lines as you want, and can mix graphs and customgraphs. The order you list them in is the order they will appear in the resultant page.

template - you specify a view template to use to generate the page. See the docs on *view template|configfile-view-templates* for explanations.

datapage - you specify a datapage to use to generate the page. See the docs on *datapage.cgi|datapage-cgi* for explanations.

You can also specify a `description` line, like:

```
desc    This is what I'm taking about
```

Where the pages are generated is dependant on the `viewdir` and `viewurl` directives in the *html config-file|configfile-html*. The view pages may have the usual indices on them, if the *html config-file|configfile-html* includes:

```
viewindices     yes
```

but by default leave them off.

View Templates

View templates give you complete control over page layout in a view. They are complete HTML pages with embedded *magic cookies|cookies*, which are substituted for during view generation (by *view-writer*). The resulting page will be an `rrdcgi` CGI script. The magic cookies are:

<VIEW::GRAPH hostname rrdname graphname [graphtime]> - This inserts a graph definition for `rrdcgi`. The graphtime is from the *times config-file|configfile-times*, and is optional.

<VIEW::CUSTOMGRAPH customgraphname [graphtime]> - This inserts a customgraph definition for `rrdcgi`. The graphtime is from the *times config-file|configfile-times*, and is optional.

<VIEW::INCLUDE filename> - This inserts the contents of the named file when the view-page is displayed. (Using the `rrdcgi` cookie <RRD::INCLUDE filename>.)

<VIEW::HEADER title here> - Inserts a standard remstats header.

<VIEW::FOOTER> - Inserts a standard remstats footer.

<VIEW::STATUS host status-file> - inserts the contents of the named status-file when the view-page is displayed. (Using the `rrdcgi` cookie <RRD::INCLUDE filename>.)

You can also include `rrdcgi` magic cookies.

Configuration Tools

These tools are intended to help you build the hosts part of your configuration file. They take a file (or files) of hostnames and emit host config-files for them. There are currently no config generators for the log-collector, the remoteping-collector or the unix-status-collector.

split-config - converts old config-files to new *config-dirs|configuration*

macinfo - show what is connected to which ports of an ethernet switch

new-config - makes a new config-dir populated by symlinks to config-base

new-ping-hosts - adds a hosts with a ping rrd

new-port-hosts - adds hosts which are collected by the *port-collector*

new-snmp-hosts - adds hosts which are collected by the *snmp-collector*

new-unix-hosts - adds hosts which are running the *unix-status-server*

nt-discover - finds and adds Windows NT hosts

snmp-get - for testing if you can get a particular OID

snmp-showif - shows interfaces from SNMP

snmpif-description-updater - like it says

update-switch-ports - like it says

macinfo

Usage:

../macinfo version 1.10 from remstats 1.0.13b usage: ../macinfo [options] [community@]host where options are:

-d ddd enable debugging output at level ‘ddd’
-h show this help
-i show IP numbers, where available
-m show MAC addresses
-n show domain-names
-N show short domain-names (before the first ".")
-u uuu show unknown name/ip/mac as ‘uuu’ [?]

Note: if community isn‘t supplied on the command-line, then it must be supplied in the environment in COMMUNITY.

Description:

This utility shows what is connected to which ports of an ethernet switch. It uses SNMP to query the bridge MIB to get MAC addresses and map them to ports. It then gets mappings from port number to ifIndex and thence to interface names (either ifName or ifDescr). Then it looks in the data from arpwatch, if available, and in /proc/net/dev to get MAC to IP# mappings. Finally, it shows what is on which port.

It‘s used by *macinfo.cgi|macinfo-cgi* to make this information available on a web-page, and *update-switch-ports* to update the descriptions of switch ports with the name of the host on that port.

new-config - make a new config-dir

Usage:

new-config version 1.17 from remstats 1.0.13b usage: new-config [options] new-config-dir where options are:

-d ddd set debug level to ‘ddd’
-f fff use ‘fff’ as config-base [/home/remstats/etc/config-base]
-h show this text

Description:

`new-config` makes a new config-dir and populates it. Things which you‘re less likely to want to change are symlink‘ed into the new config-dir. Things which you are expected to change, are copied. Look at your new config-dir to see which things are symlinks and which aren‘t.

It‘s also a good idea to check everything in the new config-dir for the string **FIXME**, which is used to indicate something which can‘t be figured out automatically by configure, but still needs to be changed. You could use a shell command like the following to get a list of files:

```
% find /new/config/dir -type f -exec grep -l FIXME {} /dev/null \;
```

new-ping-hosts - add ping RRD to host definition

Usage:

new-ping-hosts version 1.11 from remstats 1.0.13b usage: new-ping-hosts [options] group [hostfile ...] where options are:

-d nnn enable debugging output at level ‘nnn’
-f fff use ‘fff’ as config-dir [/home/remstats/etc/config]
-h show this help

Description:

You supply a list of files containing hostnames, one per line. If there is no hostfile supplied, then it will read from stdin. If there is no host config-file for that host, `make-ping-hosts` will write entries like:

```
# hosts/www.example.com
desc    gggg host
group   gggg
ip      123.456.789.123
tools   ping traceroute
```

```
rrd     ping
```

for that host. If that host already exists, it will simply add:

```
rrd    ping
```

to the end of the hosts file. It doesn‘t check if the host already has a ping rrd.

new-port-hosts - add RRDs for services

Usage:

new-port-hosts version 1.12 from remstats 1.0.13b usage: new-port-hosts [options] group [hostfile ...] where options are:

-d enable debugging output
-f fff use ‘fff’ for config-dir [/home/remstats/etc/config]
-h show this help

Description:

You supply a file or files of hostnames, one per line, or let it read from stdin.
You can use this to add RRDs to a host describing the various services running on a server, or at least the ones that the *port-collector* knows how to talk to. It‘s actually a very limited port-scanner, and will attempt to connect to each of the services. For each one that answers, `new-port-hosts` will write an entry for the corresponding rrd.
If the host has no file, `new-port-hosts` will add one with appropriate header info and a ping rrd. Otherwise, it will just add the port-based rrd‘s to the end of the host file.

new-snmp-hosts - add RRDs collected by snmp-collector

Usage:

new-snmp-hosts version 1.14 from remstats 1.0.13b usage: new-snmp-hosts [options] group community-string [hostfile ...] where options are:

-d enable debugging output
-f fff use ‘fff’ for config-dir [/home/remstats/etc/config]
-h show this help

Description:

if you don‘t supply a file of hostnames, then it will read from stdin.
You can use `new-snmp-hosts` to add RRDs collected by the snmp-collector. It works by looking for a single OID for each RRD. If the OID exists on a given host (i.e. it returns data), then the RRD which uses that data is added to the host. There is currently no way to configure it except for modifying the code. Complain if you‘d actually use such a thing.
It‘s up to you to discard any you don‘t want.
If there is no hosts file, it will create one with default header info. If there is one, it will just append the rrds.

new-unix-hosts - add rrds collected by the unix-status-collector

Usage:

new-unix-hosts version 1.12 from remstats 1.0.13a usage: new-unix-hosts [options] group [hostfile ...] where options are:

-d enable debugging output
-f fff use ‘fff’ for config-dir [/home/remstats/etc/config]
-h show this help
-p ppp use port ‘ppp’ [1957]
-t ttt use ‘ttt’ for timeout [10]

Description:

If you don‘t supply a file of hostnames, then it will read from stdin.
This will add all the remstats distributed rrds collected by the *unix-status-collector*, except for those using the PS section of the collector.
It‘s up to you to discard any you don‘t want.

If there is no existing host file for a given host, it will create one with default header info. If there is one, it will just append the new rrds.

nt-discover - find and add new NT hosts

Usage:

nt-discover version 1.8 from remstats 1.0.13b usage: ../nt-discover [options] where options are:

```
-d nnn  enable debugging output at level 'nnn'
-f fff  use 'fff' for the config-dir [/home/remstats/etc/config]
-h      show this help
-s      update status files even if host data-dir found
-t ttt  use 'ttt' as timeout (in seconds) [10]
```

Description:

Note: `Nt-discover` is not called `new-nt-hosts` because it is a different kind of program. Instead of you providing a list of hosts to add, it finds them itself.
Using information supplied in the *discovery config-file|configfile-discovery*, `nt-discover` will contact a host running the *nt-status-server*, and run three separate queries:

`NET-VIEW` will give a list of hosts to check

`USRSTAT` will give a list of NT users (currently unused, but may be interesting)

`SRVINFO` (for each of the hosts found in the first step) will give some more details on each host and write new host config-files for each new one. It will not update an existing host config-file.

It will create files under /home/remstats/data/NT to store the discovered hosts and users.

pattern-query

Usage:

pattern-query version 1.2 from remstats 1.0.13b usage: ../pattern-query [options] pattern-file [file-to-search] where options are:

```
-h      show this help
-m mmm  check 'mmm' matches at most [10]
```

Description:

This is for testing the `infopattern` and `valuepattern` regex's used in the *port-collector* scripts. Usually it's used in conjunction with the *port-query* program. The `port-query` program fetches data and the `pattern-query` program tests that data against various patterns.

port-query

Usage:

port-query version 1.6 from remstats 1.0.13b usage: ../port-query [options] host port [arg] ... where the optional args are to be substituted for ##1##, ##2##, ... where options are:

```
-d nnn  enable debugging output at level 'nnn'
-f fff  read query from file 'fff' [stdin]
-h      show this help
-s      do magic-cookie substitutions on query
-t ttt  use 'ttt' for timeout for response [10]
```

Description:

Sort of like `telnet host port`, the `port-query` program connects to the named host on the specified port, sends the contents of the query file and returns whatever the service on the other end says. Can be used with the *pattern-query* program to test scripts used by the *port-collector*.

remstats-rrdtune

Usage:

remstats-rrdtune version 1.4 from remstats 1.0.13b usage: ../remstats-rrdtune [options] where options are:

```
-d nnn  enable debugging output at level 'nnn'
-h      show this help
        -G GGG  only apply to hosts in groups 'GGG' (a comma-separated list)
```

-H HHH only apply to hosts 'HHH' (a comma-separated list)

Note: Only one -H or -G may be used in making a key. If -G is used with -H, the keys will select a subset of the hosts selected by -H or -G.

Description:

The purpose of `remstats-rrdtune` is to synchronize the remstats RRD definitions from the *rrds config-dir|configfile-rrds* with the actual RRD files. It can only apply changes which `rrdtool tune` would let you make, so changing RRAs cannot be done, e.g. You can change the ds-type, heartbeat, minimum and maximum.

snmp-get

Usage:

snmp-get version 1.11 from remstats 1.0.13b usage: snmp-get [options] {-c ccc | -r rrr} host oid where options are:

-d ddd set debugging output to level 'ddd'
-h show this help
-f fff use 'fff' for config-dir [/home/remstats/etc/config]
-p ppp use port 'ppp' instead of the usual port

required args are only one of:

-c ccc use 'ccc' as SNMP community-string
-r rrr get SNMP community-string from RRD 'rrr'

- 'host' is the host to query - 'oid' is the Object ID to return the value of, either fully numeric or as defined in the 'oids' config-file

Description:

Gets a value via SNMP. In case you don't have another snmp package.

snmp-showif - display interfaces from SNMP

Usage:

snmp-showif version 1.8 from remstats 1.0.13b usage: ../snmp-showif [options] host community where options are:

-d enable debugging output
-h show this help

Description:

This utility can be used to get a list of interfaces on an SNMP queryable host. You can use it to figure out which interfaces you want to add, and what names remstats uses for them.

It will show ifIndex, ifDescr, ifSpeed, ifType, ifName, ifInOctets, ifOutOctets, ifOperStatus and ifAlias (if it exists).

snmpif-description-updater

Usage:

snmpif-description-updater version 1.11 from remstats 1.0.13b usage: snmpif-description-updater [options] where options are:

-d ddd set debugging level to 'ddd'
-f fff use 'fff' for config-dir [/home/remstats/etc/config]
-h show this help message

Description:

This script updates host config-files, for those which contain `snmpif-*` RRDs, by fetching the ifAlias OID via SNMP and re-writing the configuration file if any of the descriptions have changed. I'd suggest running it every now and then, say once a day, unless you're making very frequent changes to the descriptions.

If somebody has gear which uses an OID other than ifAlias to store the description, then I'll have to consider making this more general, but it'll do for now.

update-switch-ports

Usage:

update-switch-ports version 1.5 from remstats 1.0.13b usage: ../update-switch-ports [options] where options are:

-d nnn enable debugging output at level ‘nnn’
-f fff use ‘fff’ as config-dir [/home/remstats/etc/config]
-h show this help

Description:

This runs *macinfo* on each host with a `switch` tag. From the reported names it updates the description for each port rrd on the switch to contain the name of the host on that port, shown as `{hostname}`. It will replace any previous host noted by an earlier run and will append to an existing description.

Servers

There are a four servers currently:

unix-status-server is queried by the *unix-status-collector* for various information it obtains by running various commonly-available programs: `df`, `vmstat`, `uptime`, `netstat`, `uname`, `ps` ...

log-server (queried by the *log-collector*) reads the unread portion of the specified log-file and returns the requested statistics.

remoteping-server is contacted by the *remoteping-collector* which supplies a list of hosts. The server runs *multiping* against them and returns results for each, similar to the results obtained from the *ping-collector*.

nt-status-server - provides access to information from Windows NT workstations and servers.

log-server - providing remote access to log information

Usage:

log-server version 1.15 from remstats 1.0.13b usage: ../log-server [options] logfile ... where options are:

-d nnn enable debugging output at level ‘nnn’
-p ppp set the prefix for context-files to ‘ppp’ [log-server-]
-h show this help

The log-server must be supplied with at least one log-file to serve.

Description:

The log-server is queried by the *log-collector* using a "protocol" described in the log-collector documentation. It will provide information from any of the log-files on it‘s command-line, but no others. It is recommended that you use the tcp_wrappers or some other form of access-control to limit access to this server. The information may or may not be sensitive, according to which log-files you are serving, but letting anyone query it will mean that you will lose some data, unless you‘re sure that they will only query it in test mode.

The log-server will store context for each log-file that is served, by default in `/var/tmp/log-server-XXX`, where `XXX` is replaced by a munged version of the log-file name. If you want this stored somewhere else, use the `-p` switch or change the program.

Notes:

Don‘t forget to list all the log-files that you want to serve on the command-line. If there are too many for your inetd, make a tiny shell script with the `log-server` invocation and run that from inetd.

For details on installation, you‘d better look at the *server installation docs|install-servers*.

nt-status-server - allow remote gathering of Windows NT data

Usage:

```
usage: nt-status-server [options]
where options are:
    -a      show all available performance counters
    -d ddd  enable debugging at level 'ddd' [$main::debug]
    -h      show this help message
    -i [ppp sss] install this as an NT service, using 'ppp' as
            as perl and 'sss' as this script.  Defaults to
                perl=C:\Perl\bin\perl.exe
                script=wherever-it's-invoked-from
    -p ppp  run server on port 'ppp' [1957]
```

```
-s      run stand-alone, i.e. not as a service
    -t ttt  trust hosts 'ttt' (comma-separated) [127.0.0.1]
    -T      show status in title-bar of window
-u      un-install this service
```

Description:

The nt-status-server allows the *nt-status-collector* to get data from a remote machine running some flavour of NT and possibly Windows 2000. It runs various programs from the NT Resource Kit, for some information and examines the NT performance counters for other information.
For details on installation, look at the *server installation docs|install-servers*.

Protocol:

The *nt-status-collector* connects to the `nt-status-server` and sends a series of commands, ending with 'GO'. Then the server sends back the data it obtained and closes the connection. The commands are often the names of programs to run (in UPPERCASE) and the ones known currently are:

SRVINFO - runs SRVINFO and returns the version of NT

PERFCOUNTERS - examines the NT performance counters and returns information about memory, disk, processes, ...

PULIST - runs PULIST (from the NT ResKit) and shows counts for all the running processes.

MSDRPT - runs WINMSDP to show (currently) memory total and free.

USRSTAT - runs USRSTAT (from the NT ResKit) and shows when the various users in the specified NT domain last logged in, and which domain-controller authorized them.

NETVIEW - runs "NET VIEW" to list the computers currently visible.

TIME - compares local and remote times

If you want to see what it returns, you can simply start it up and telnet to it.

Bugs

It is intended that it will eventually install itself as an NT service, and most of the code is there, but it doesn't currently work. Patches gratefully accepted. For now you have to invoke it with the `-s` switch to have it run stand-alone or use SRVANY to provide the NT service stuff, (which also requires the `-s` flag.

Not only is is currently single-threaded (i.e. won't accept more than one connection at a time), but if a second connection comes in the server won't answer any more requests and will have to be re-started. I've added code to *nt-status-collector* so that it won't run if there is another instance running already. This won't help if you're using telnet to test the nt-status-server, so be prepared to restart nt-status-server if it gets wedged because of this.

remoteping-server - allow remote collection of ping data

Usage:

remoteping-server version 1.7 from remstats 1.0.13b usage: ../remoteping-server [options] where options are:

-d nnn	enable debugging output at level 'nnn'
-h	show this help

Description:

The remoteping-server allows the *remoteping-collector* to obtain ping data remotely. Like the *ping-collector*, it uses *multiping|multiping.html* to get ping data, but it can be queried from a remote site.
I'm looking for volunteers; please look at *this note|remoteping-collector.html#note*.

unix-status-server - allow remote gathering of unix data

Usage:

unix-status-server version 1.43 from remstats 1.0.13a usage: ../unix-status-server [options] where options are:

-d nnn enable debugging output at level 'nnn'
-h show this help
-r include remotely-mounted file-systems
-t tst do tests 'tst', a comma-separated list of:

VMSTAT, DF, UPTIME, NETSTAT, UNAME, PS, PROC, MASQCONN, FTPCOUNT, NETSTAT-TCPSTATES, FILEAGE, QMAILQSTAT, QMAILQREAD, PROCNAME, PROCDISKIO, PROCMEMINFO, SAR

Description:

The unix-status-server allows the *unix-status-collector* to get data from a remote machine running some flavour of unix. It runs several different programs on request (uname, vmstat, df, uptime, netstat, ps, ftpcount, qmail-qstat, and qmail-qread).

Protocol:

The *unix-status-collector* connects to the `unix-status-server` and sends a series of commands, ending with 'GO'. Then the server sends back the data it obtained by running the requested programs and closes the connection. The commands are usually the names of programs to run (in UPPERCASE) and the ones known currently are:

UNAME runs uname and returns: machine, os_name, os_release, os_version

VMSTAT runs vmstat and returns variables relating to memory usage depending on the operating system

DF runs df and for each file-system returns: dfsize:FSNAME, dfused:FSNAME, dfpercent:FSNAME inodessize:FSNAME, inodesused:FSNAME, inodespercent:FSNAME

UPTIME runs uptime and returns: uptime (a timestamp in seconds), users, load1, load5, load15

NETSTAT runs netstat and, for each interface, returns: interface_packets_in:IFNAME, interface_errors_in:IFNAME, interface_packets_out:IFNAME, interface_errors_out:IFNAME, interface_collisions:IFNAME

PS runs ps and returns various numbers pulled out of the output (see below)

FTPCOUNT runs ftpcount (from wuftpd) to find out which groups the ftp-server's users fall into

QMAILQSTAT runs qmail-qstat and returns: qmail_qsize, qmail_qbacklog

QMAILQREAD runs qmail-qread and returns FIXME

FILEAGE returns timestamps for the ages of specified files (see below)

TIME return the difference in time-stamps between the host running the unix-status-server and the querying host. It must be given the querying host's timestamp following the `TIME` directive. The two variables returned are time and timediff.

NETSTAT-TCPSTATES runs netstat and produces counts of the number of tcp sessions in the various TCP states.

SAR runs "`sar -A`" and produces a lot of operating-system specific info.

If you want to see what it will return, you can simply invoke the `unix-status-server` as a local script and type commands at it.

Programs:

`/usr/local/bin/uname` or `/usr/bin/uname` It's mostly, cosmetic, for web-page header info, but sometimes it's really usefull too. It's required for some of the other sections, in order to figure out how to adapt to various operating-system sepcific quirks.

`/usr/bin/vmstat` or `/usr/ucb/vmstat` for scanrate, interrupts, context-switches and cpu-time, and of course free memory and swap

`/usr/local/bin/df` or `/usr/xpg4/bin/df` or `/bin/df` the gnu df. I need the -P flag (for Posix, but it makes df put its info on one line), and the -i flag for inode info.

`/usr/local/bin/uptime` or `/usr/bin/uptime` or `/usr/ucb/uptime` the gnu uptime has a format I know how to parse. Others often invent new ways to be cute about the display that I don't always recognise. Please show me examples of formats that the unix-status-server doesn't know, and which os (uname -a) they come from.

`/usr/bin/netstat` or `/usr/ucb/netstat` to get network interface info

`/usr/bin/ps` or `/bin/ps` for counting the number of running instances of a named process.

`/usr/local/bin/ftpcount` (part of wu-ftpd distribution) shows the number of ftp clients of wu-ftpd from each access group.

`/var/qmail/bin/qmail-qstat` and `/var/qmail/bin/qmail-qread` If you have *qmail|http://www.qmail.org/*, these will let you get information about the queue size, which you

can't find from the logs. Otherwise, ignore them.

PS Usage:

With extended commands, of which PS is the first, you also specify what you want to look for with extra commands, in addition to the PS command. A command looks like:

```
varname PS func pattern
```

The `varname` is used to create a variable-name for the returned data. The name will be `ps:varname`. `Func` is one of `count`, `sum`, `last`, `average`, `min`, `max`. `Pattern` is a perl-style regular-expression, the simplest form of which is just a string.
For an example, if we wanted to know how many web-servers were running over time, we might use (very sloppily):

```
webservers PS count httpd
```

[You probably want a better regular expression.]

FILEAGE Usage:

For the `FILEAGE` command, you have to specify an extended command that looks like:

```
varname FILEAGE /path/to/file
```

This will produce a timestamp for the last-modification time of `/path/to/file`.

PROCNAME Usage:

PROCNAME examines /proc (only under linux) and produces counts of specified named processes

PROCDISKIO Usage:

PROCDISKIO examines /proc/stat (only under linux) and returns disk I/O operation and bytes counts.

PROCMEMINFO Usage:

PROCMEMINFO examines /proc/meminfo (only under linux) and returns returns the memory stats. Some-one please tell me *terskine@users.sourceforge.net|mailto:terskine@users.sourceforge.net* if you can interpret these to tell when a host is **really** running out of memory.

PROCNETDEV Usage:

PROCNETDEV examines /proc/net/dev (only under linux) and returns usefull network counters

Notes:

With older versions of `vmstat` (ones that mash fields together), it will give up on `vmstat` and not return memory and CPU info. It also requires a version of `df` that will accept the `-P` and `-i` flags. The `-P` flag forces the output for a file-system to stay on one line (easier for me to parse) and the `-i` returns info about inodes. If the `-i` flag is missing, you won't get any inode data. You also won't get any inode data if the file-system doesn't have inodes. (Duh :-).
For details on installation, look at the *server installation docs|install-servers*.

Collectors Data Format

All the collectors produce data on stdout in the same standard form:

```
host timestamp variable value
```

If the `variable` is for something like network interfaces, where the host can have several of them, the data will look like:

```
host timestamp variable:instance value
```

Having all the collectors using a standard form permits a single updating program, *updater*, to process the data from them all, and also means that I can write a new collector without needing to change the updater.

Remstats supplied collectors

log-collector|log-collector.html - gets info from remote log-files

ping-collector - pings hosts

port-collector - checks on remote services

program-collector - runs arbitrary commands to collect info

remoteping-collector - pings hosts from somewhere else

unix-status-collector - gets info from unix hosts

snmp-collector - gets info via SNMP

snmp-route-collector - counts routes available from BGP peers

The usual invocation of a collector (via *run-remstats*) is:

```
xxx-collector | updater xxx
```

How to write your own collector

There are a few requirements:

1) it must write its results to stdout in standard form (see the top of this file.)

2) it must be placed in the same directory with the rest of the collectors, and must be called XXX-collector, replacing "XXX" with whatever it's collecting.

3) it must take (or at least ignore), the same arguments that the other collectors do, specificly, the `-f`, `-u` and `-F` flags, and the `-G` and `-H` flags, if I ever get around to implementing them (see *todo*).

4) you must add it to the list of collectors (the `collectors` line in the *general config-file |configfile-general*).

5) you must define rrd(s) specifying "source XXX" to use the data from this collector.

6) you must add "rrd YYY" to the appropriate *host config-files|configfile-hosts*.

There is a supplied `skeleton-collector.pl` file supplied with the distribution, which does everything except collect data. You should be able to plug your code into its `collect_host` routine and have a collector, if you don't mind writing in perl.

cisco-access-server-collector

Usage:

cisco-access-server-collector version 1.21 from remstats 1.0.13b usage: ../cisco-access-server-collector [options] where options are:

```
-c ccc  use 'ccc' for the read community string; overrides host
-d nnn  enable debugging output at level 'nnn'
-f fff  use 'fff' for config-dir [/home/remstats/etc/config]
-F      force collection even if it's not time
        -G GGG  only try hosts in groups 'GGG', a comma-separated list
-h      show this help
-H HHH  only try hosts from 'HHH', a comma-separated list
        -K KKK  only try hosts with key(s) 'KKK', a comma-separated list
-u      ignore uphosts file
```

Description:

Collects, via SNMP, totals from a Cisco Access Server 52xx or 53xx because the *snmp-collector* doesn't do roll-ups.

dbi-collector - data from remote databases

Usage:

dbi-collector version 1.8 from remstats 1.0.13b usage: ../dbi-collector [options] where options are:

```
-d nnn  enable debugging output at level 'nnn'
-f fff  use 'fff' for config-dir [/home/remstats/etc/config]
-F      force collection even if it's not time
        -G GGG  only try hosts from group 'GGG', a comma-separated list
-h      show this help
-H HHH  only try hosts from 'HHH', a comma-separated list
-K KKK  only try hosts with key(s) 'KKK', a comma-separated list
-u      ignore uphosts file
```

Description:

The `dbi-collector` uses perl's DBI interface to access remote databases to collect data. If you have a DBD driver for your database, and can write an SQL select statement to select the data you're interested in, the `dbi-collector` can import the data into rrdtool and remstats.

The `dbi-collector` adds three new directives to RRD definitions which use it:

connect CONNECTNAME - specifies the connection to use, documented under the *dbi-connects config-dir|configfile-dbi-connects*.

select SELECTNAME - specifies the select statement to use, by name as documented in *dbi-selects config-dir|configfile-dbi-selects*.

multirowid COL# - specifies the column number to use to construct the variable name for variables from this row. This permits the use of select statements which return more than one row.

The `connect` directive tells how to connect to the appropriate server and database; the `select` tells how to select the data.

For an RRD collected by the `dbi-collector`, the `data` directive will require the column number in the select that will provide its data. This will be appended to the `data` directive like:

```
data xxx=yyy GAUGE:600:0:U COL=12
```

There are also pseudo-columns called `STATUS` and `RESPONSE` to enable the rrd definition to reference the connection-status and response-time.

There are also some possible additions to the `rrd` directive in the host file. The rrd directive can look like:

```
rrd     rrdname CONNECT="cc" SELECT="ss" USER="uu" PASSWORD="pp" DATABASE=
```

All of the extras on the end are optional, and are provided in order to be able to use a single connect file with multiple databases, users, ...

If the `multirowid` is specified, then the variable returned is named `RRDNAME:DATANAME:ROWNAME`. Otherwise, it is named `RRDNAME:DATANAME`.

Notes

The `dbi-collector`, when querying an Oracle database, needs to know where Oracle was installed, specifically, it needs the environment to contain `ORACLE_HOME` pointing at where Oracle was installed. This can be simply done using the new *environment config-file|configfile-environment*.

dns-collector

Usage:

dns-collector version 1.2 from remstats 1.0.13b usage: ../dns-collector [options] where options are:

```
-d nnn  enable debugging output at level ‘nnn’
-f fff  use ‘fff’ for config-dir [/home/remstats/etc/config]
-F      force collection even if it‘s not time
        -G GGG  only try hosts from group ‘GGG‘, a comma-separated list
-h      show this help
-H HHH  only try hosts from ‘HHH‘, a comma-separated list
-K KKK  only try hosts with key(s) ‘KKK‘, a comma-separated list
-u      ignore uphosts file
```

Description:

The `dns-collector` does DNS queries to test availability of specified records and response time of the server providing them.

The queries are made to the host that the DNS rrds are defined on. The records are specified in the rrd instance. E.G.

```
rrd dnssite-example.net SOA A NS MX
```

This example will query the host that it belongs to for `SOA`, `A`, `NS`, and `MX` records for `example.net`.

error-collector

Usage:

error-collector version 1.1 from remstats 1.0.13b usage: ../error-collector [options] run-pid where options are:

```
-d nnn  enable debugging output at level ‘nnn’
-f fff  use ‘fff’ for config-dir [/home/remstats/etc/config]
-F      force collection even if it is not time
        -G GGG  only try hosts from group ‘GGG‘, a comma-separated list
-h      show this help
```

-H HHH only try hosts from ‘HHH‘, a comma-separated list
-K KKK only try hosts with key(s) ‘KKK‘, a comma-separated list
-u ignore uphosts file

Description:

More remstats self-monitoring. It counts the errors and aborts produced by the other collectors.

log-collector - get stats from remote log-files

Usage:

log-collector version 1.21 from remstats 1.0.13b usage: ../log-collector [options] where options are:
-d nnn enable debugging output at level ‘nnn’
-f fff use config-dir ‘fff‘[/home/remstats/etc/config]
-F force collection, even if it is not time
-G GGG only do hosts from group ‘GGG‘, a comma-separated list
-h show this help
-H HHH only do hosts from ‘HHH‘, a comma-separated list
-K KKK only try hosts with key(s) ‘KKK‘, a comma-separated list
-p ppp contact log-server on port ‘ppp’ [1958]
-t ttt timeout each port attempt after ‘ttt’ seconds [10]
-u ignore uphosts file

Description:

The log-collector gets data from remote *log-server*‘s. This way the whole log-file doesn‘t have to be transfered. The "protocol", if it deserves that name, is very simple. The collector sends a request, which looks like (you can type it in via telnet):

```
LOGFILE /wherever/the/logfile.is
varname function pattern
...
GO
```

The directives are all in **UPPERCASE**. They are `LOGFILE`, `GO`, `DEBUG` and `TEST`. The `LOGFILE` directive tells the `log-server` which file to read. The `GO` directive starts the request. `DEBUG` causes some extra remote debugging output, and `TEST` makes the `log-server` operate in test mode. In test mode it doesn‘t update the last-read position for that log-file, so you won‘t lose any data when testing.
The other lines are telling the `log-server` what data to collect. The first "word" is the variable name to be returned. The next is the function to be applied (from `count`, `sum`, `average`, `min`, `max`, `first`, `last`) and `lastof`. The rest of the line is a perl-style regex.
Note1: The `lastof` function is different from the others, in several respects. First, it takes multiple regex‘s. Second, it saves the current value of the variable in a context file with the same name as the context file for the log-file with ‘-’ and the variable-name appended. The way it works is to return the number of the pattern which was last matched (starting with 1). If none of the patterns have ever matched, it will return zero, until one of them matches.
Note2: Except for the count function, the regex must contain a (parenthesized) expression which will match a number, to which the function will be applied.
For example, the line:

```
rootlogins count ROOT LOGIN
```

would return data for a variable called rootlogins. The value would be the count of the records in the specified logfile which had the string ‘ROOT LOGIN’ in them.
The pattern can be much more complicated, for example (from the httpdlog rrd):

```
bytes sum \sHTTP/\d\.\d"\s+2\d\d\s+(\d+)
```

This looks through a standard web-server log-file and extracts the bytes transferred and adds them up to produce the total number of bytes transferred for successes in that sample period.

How to make RRDs that use the log-collector

It‘s easiest to explain by example. Look at the beginning of the rrd `httpdlog`, copied here:

```
source  log
step    300
data    requests        GAUGE:600:0:U   count (GET|POST)
```

```
data    success         GAUGE:600:0:U   count \sHTTP/\d\.\d"\s+2\d\d
data    bytes           GAUGE:600:0:U   sum \sHTTP/\d\.\d"\s+2\d\d\s+(\d+)
```

To form the requests to be sent to the log-server, the log-collector takes the DS name, e.g. `success`, and the last part of the line after all the DS definition `count \sHTTP/\d\.\d"\s+2\d\d`, combines the two and sends:

```
success count \sHTTP/\d\.\d"\s+2\d\d
```

Note that the pattern can include *magic cookies|cookies* as of remstats version 0.12.2.
The line in the host config-file must specify the log-file name, like:

```
httpdlog /var/log/httpd/access_log
```

This permits the same code to get information from the same log format, no matter where it‘s stored on various machines. Since log-file locations seem to vary a lot by host, I put the log-file specification in the host config-file, rather than in the rrd config-file.

nt-status-collector - stats from Windows NT hosts

Usage:

nt-status-collector version 1.40 from remstats 1.0.13a usage: ../nt-status-collector [options] where options are:

-d nnn enable debugging output at level ‘nnn’
-f fff use ‘fff’ for config-dir [/home/remstats/etc/config]
-F force collection, even if it is not time
-G GGG only try hosts from group ‘GGG‘, a comma-separated list
-h show this help
-H HHH only try hosts from ‘HHH‘, a comma-separated list
-p ppp connect to server on port ‘ppp’ [1957]
-t ttt set timeout to ‘ttt’ [25]
-K KKK only try hosts with key(s) ‘KKK‘, a comma-separated list
-u ignore uphosts file

Description:

The `nt-status-collector` gets its data from the *nt-status-server*. It sends a query consisting mostly of the sections of the server that it wants to run. It can also request that processes with specific names be counted and that information returned too. A query for all the sections might look like:

```
SRVINFO
PERFCOUNTERS
PULIST
MSDRPT
USRSTAT ntdomain
NET-VIEW
GO
```

Note that `PULIST`, `MSDRPT`, `USRSTAT` and `NET-VIEW` aren‘t currently used by anything, but they may be usefull for something. Also, USRSTAT wants an NT domain-name with the query.

ntop-collector

Usage:

ntop-collector version 1.9 from remstats 1.0.13b usage: ../ntop-collector [options] where options are:

-d nnn enable debugging output at level ‘nnn’
-f fff use ‘fff’ for config-dir [/home/remstats/etc/config]
-F force collection even if it is not time
-G GGG only try hosts from group ‘GGG‘, a comma-separated list
-h show this help
-H HHH only try hosts from ‘HHH‘, a comma-separated list
-K KKK only try hosts with key(s) ‘KKK‘, a comma-separated list
-u ignore uphosts file

Description:

To be written.
FIXME

ping-collector - get reachability of hosts

Usage:

ping-collector version 1.25 from remstats 1.0.13b usage: ping-collector [options] where options are:
-d nnn enable debugging output at level ‘nnn’
-f fff use ‘fff’ for config-dir [/home/remstats/etc/config]
-F force collection even if it is not time
-G GGG only try hosts from group ‘GGG‘, a comma-separated list
-h show this help
-H HHH only try hosts from ‘HHH‘, a comma-separated list
-K KKK only try hosts with key(s) ‘KKK‘, a comma-separated list
-u for compatibility with run-remstats; ignored

Description:

ping-collector uses *multiping* to get numbers on how reachable the hosts are. Each host is sent 10 pings (ICMP echo-request) and the number of responses and the min, max and average RTT (Return Trip Time) is logged, giving the variables ping-sent, ping-rcvd, pingrtt-min, pingrtt-avg and pingrtt-max.

multiping

Usage:

/bin/sh: line 1: ../multiping: is a directory

Description:

Multiping runs pings in parallel which permits you to ping lots of hosts quickly. I wrote a program using Net::Ping to try to how bad a simple-minded approach was. With .025s between pings in my programs and 1s in multiping, my program took 17 minutes and multiping took 37 seconds. Maybe someone will not be able to get multiping going and will re-write it in perl. Not me. I‘d be happy to include it if some-one wants to send me a copy.

port-collector - get service status

Usage:

port-collector version 1.26 from remstats 1.0.13b usage: ../port-collector [options] where options are:
-d nnn enable debugging output at level ‘nnn’
-f fff use ‘fff’ for config-dir [/home/remstats/etc/config]
-F force collection even if it is not time
-G GGG only try hosts from group ‘GGG‘, a comma-separated list
-h show this help
-H HHH only try hosts from ‘HHH‘, a comma-separated list
-t ttt set default timeout to ‘ttt’ [5]
-K KKK only try hosts with key(s) ‘KKK‘, a comma-separated list
-u ignore uphosts file

Description:

The `port-collector` gets data about services running on specified TCP ports. It will attempt to connect to the specified port on the host, and optionally send a string to it. It will then examine the response, if any, for certain patterns. This permits it to query almost any text-based protocol. For information on how to set up a new service, look in the *scripts|configfile-scripts* directory in the configuration directory.
The rrd specification can also override the port specified by the script, by appending a colon and the port-number. E.G. for a web-server on port 8000, you could specify the rrd like:

```
rrd port-http:8000
```

According to whether it can connect to the service and what response it gets back, it sets a status to one of OK, WARN, ERROR, CRITICAL. These are arbitrary levels, except that OK means normal, and their meanings are determined by the configuration file. The `port-collector` will also log how long it took the service to respond. These numbers are not intended for benchmarking, but only for

determining the health of the service.

Returned Data

The variables returned by the port-collector are:

```
port-PORTNAME - containing the status of the port, calculated by
        the script for this port
port-PORTNAME-response - containing the response-time for the query
```

Other data from the port-collector

The main RRD for the port-collector is `port-*`, but it is possible to define other RRDs. If you want to collect information from the results elicited by the send string, you can provide either or both of the `infopattern` or `valuepattern` in the script associated with the RRD. The script must be named the same as the rrd. Look in the *scripts configfile docs|configfile-scripts* for details on scripts.

A matching `valuepattern` will cause the port-collector to return variables named `RRDNAME:value#` with "#" replaced by a single digit, corresponding to the number of the parenthesized part of the pattern that was matched.

A matching `infopattern` will cause the port-collector to create status files for this host called INFO1-RRDNAME. None of the existing *pagemakers* use these status files, but the *view-writer* could do so if a *view template|configfile-view-templates* refered to them.

program-collector

Usage:

program-collector version 1.5 from remstats 1.0.9b usage: ../program-collector [options] where options are:

```
-d nnn  enable debugging output at level ‘nnn’
-f fff  use ‘fff’ for config-dir [/home/remstats/etc/config]
-F      force collection even if it is not time
    -G GGG  only try hosts from group ‘GGG‘, a comma-separated list
-h      show this help
-H HHH  only try hosts from ‘HHH‘, a comma-separated list
-K KKK  only try hosts with key(s) ‘KKK‘, a comma-separated list
-u      ignore uphosts file
```

Description:

The `program-collector` runs a configuration-specified program to get the values for its RRD. The program must print its output in one of two forms.

The simple form is just simple values, one per line. The collector will invent names for each value, based on the RRD name. For the first value, the name for the value will be the name of the RRD. Any subsequent values will be named the RRD name followed by a hyphen and a number, giving names like `rrdname-2`, `rrdname-3`, ...

In the other form, the program must provide names itself, so each line is the name of the value followed by white-space and the value itself. If the values produced by the program contain white-space, the program **must** use this form.

In either case, the collector will materialize values for the response-time in named the same as the value, with "`-response`" appended. The response-time is how long the program took to produce that value, cumulative since the beginning of the running of the program.

There is supplied an RRD definition, `program-*`, to make this easier to use for simple cases. All it needs is a "program" which prints a single value. The "program" could be anything, including a pipeline. The RRD has two variables, `response` and `response-time`, the former being the value supplied by the program, and the latter the run-time of the program. E.G. to get a count of th enumber of processes running, you could include an RRD directive like:

```
rrd     program-processes /usr/bin/ps -e | wc -l
```

or, if you‘ve got a BSD version of ps:

```
rrd program-processes /usr/bin/ps ax | wc -l
```

remoteping-collector - reachability from other sites

Usage:

remoteping-collector version 1.18 from remstats 1.0.13b usage: ../remoteping-collector [options] where options are:

-d nnn enable debugging output at level ‘nnn’
-f fff use ‘fff’ for config-dir [/home/remstats/etc/config]
-h show this help
-p ppp use port ‘ppp’ instead of the default [1959]
-t ttt use ‘ttt’ for timeout [60]
-u for run-remstats compatibility

Description:

The `remoteping-collector` is intended to gather ping statistics (see *ping-collector*) from remote sites. It works by contacting *remoteping-server*s running on other machines. This way it will be possible to monitor the same list of hosts from multiple points on the network and determine several things:

general health of parts of the network of interest to the co-operating parties, and
if certain parts of the network are performing better or worse than others.

Note:

Note the use of the future tense in the previous paragraph. I‘m looking for volunteers to run the *remoteping-server* and let me have access to it. In return, I‘ll let you look at the stats that this process gathers. I‘m planning to monitor ISPs and other sites across Canada, as well as commonly accessed sites around the world, to determine how we‘re doing in network performance, here in Canada. If you want to volunteer, please hit the mailto URL below and ignore the bounce from *my spam protection|http://silverlock.dgim.crc.ca/~terskine/qmail/tms.html*.

snmp-collector - get data via SNMP

Usage:

snmp-collector version 1.28 from remstats 1.0.13a usage: snmp-collector [options] where options are:
-c ccc use ‘ccc’ for the read community string; overrides host
-d nnn enable debugging output at level ‘nnn’
-f fff use ‘fff’ for config-dir [/home/remstats/etc/config]
-F force collection even if it‘s not time
-G GGG only try hosts from group ‘GGG‘, a comma-separated list
-h show this help
-H HHH only try hosts from ‘HHH‘, a comma-separated list
-K KKK only try hosts with key(s) ‘KKK‘, a comma-separated list
-u ignore uphosts file

Description:

The snmp-collector collects data available via SNMP. There are some things that are hardcoded in, but it‘s mostly configurable. It will attempt to query for the following, if available:

sysDescr - tells what kind of a device this is
sysUptime - how long it has been up
and use them in the host index page.
The `snmpif-*` rrd is wired into the snmp-collector (for now) and causes it to fetch the following for each interface:

ifType - interface type
ifOperStatus - operational status
ifSpeed - interface speed
ifInErrors - input errors
ifOutErrors - output errors
ifInOctets - input octets (aka bytes)
ifOutOctets - output octets (aka bytes)
ifInUcastPkts - input unicast packets
ifOutUcastPkts - output unicast packets
ifInNUcastPkts - input non-unicast (broadcast and multicast) packets
ifOutNUcastPkts - output non-unicast (broadcast and multicast) packets

The sysDescr and sysUptime are saved for the host display and the ifType and ifSpeed are combined to give a hardware description for the interface. Crude, but portable.
For other SNMP data, you'll need to look at the *[oids]|configfile-oids* file in the configuration directory. The rrd will need to contain `oid` lines specifying names assigned in the *oids section|configfile-oids*. If the host doesn't have one, the rrd will also need to specify a `community`. Here's an example:

```
[rrd snmpmem]
source          snmp
step            300
data            freemem=ciscofreemem GAUGE:600:0:U
data            totalmem=ciscototalmem GAUGE:600:0:U
archives        day-avg week-avg month-avg year-avg
times           day yesterday week month year
oid             CiscoFreeMem
oid             CiscoTotalMem
```

This rrd definition will fetch the amount of free memory and total memory available on a Cisco router. Since it's querying a Cisco-specific MIB, it's not usefull on other gear.

snmp-route-collector

Usage:

snmp-route-collector version 1.22 from remstats 1.0.13b usage: snmp-route-collector [options] where options are:

-c ccc use 'ccc' for the read community string; overrides host
-d nnn enable debugging output at level 'nnn'
-f fff use 'fff' for config-dir [/home/remstats/etc/config]
-F force collection even if it's not time
-G GGG only try hosts from group 'GGG', a comma-separated list
-h show this help
-H HHH only try hosts from 'HHH', a comma-separated list
-K KKK only try hosts with key(s) 'KKK', a comma-separated list
-u ignore uphosts file

Description:

This is a specialized form of SNMP collector which walks a part of the BGP4 MIB, specificly `bgp4PathAttrBest`, to count routes available from a given BGP peer host. It notes both the total number of routes and how many of them are the best route to that destination.
Unfortunately, it doesn't scale. On routers with large numbers of peers it can take a **long** time to troll through all the routes. This needs to be replaced.

unix-status-collector - stats from unix hosts

Usage:

unix-status-collector version 1.31 from remstats 1.0.13b usage: unix-status-collector [options] where options are:

-d nnn enable debugging output at level 'nnn'
-f fff use 'fff' for config-dir [/home/remstats/etc/config]
-F force collection even it it is not time yet
-G GGG only try hosts from group 'GGG', a comma-separated list
-h show this help
-H HHH only try hosts from 'HHH', a comma-separated list
-K KKK only try hosts with key(s) 'KKK', a comma-separated list
-p ppp connect to server on port 'ppp' [1957]
-t ttt set timeout to 'ttt' [10]
-u ignore uphosts file

Description:

The unix-status-collector gets its data from the *unix-status-server*. For a detailed explanation of what the various directives mean, see its documentation, as it's the implementor. It sends a query consisting mostly of the sections of the server that it wants to run. It can also request that processes with specific

names be counted and that information returned too. A query for all the sections might look like:

```
UNAME
UPTIME
TIME 986485967
VMSTAT
DF
NETSTAT
QMAILQ
PS
webservers ps count httpd
FILEAGE
test fileage /var/spool/locks/lockfile
PROC
swaptot proc /proc/meminfo ^SwapTotal:\s+(\d+)
GO
```

The `webservers ps count httpd` line requests that the ps section count the number of processes called `httpd` and return that as a variable called `webservers`.
The `test fileage /var/spool/locks/lockfile` line requests the last modification time of the file `/var/spool/locks/lockfile`, which is returned in seconds.
The `swaptot ...` line looks for the total swap size in `/proc/meminfo`.
The best way to see what it will produce is to run it manually.

updater - add new data to RRDs

Usage:

updater version 1.23 from remstats 1.0.13b usage: updater [options] collector where options are:
-d nnn enable debugging output at level ‘nnn’
-f fff use ‘fff’ for config-dir [/home/remstats/etc/config]
-h show this help
-H hhh only process host ‘hhh’
-r rrr update rrd file ‘rrr’ (see note1)
-R RRR update only rrd ‘RRR’
Note1: The -r option must be combined with the -H and -R options and is intended only to be used by rt-updater. It will take data only for the specified host (-H) and RRD (-R) and update the specified (-r) RRD file, instead of the usual calculated one.

Description:

It reads collector output in standard form from stdin and updates the appropriate RRDs. It wants to know which collector the information came from to avoid looking for information that won‘t be there.
It is responsible for figuring out which value belongs to which RRD and variable. It is also responsible for applying any functions to that value. It also implements the FAKE function, see the docs for the *rrds config-dir|configfile-rrds*.

Monitors

Currently, there are three monitors:

ping-monitor - determines reachability of hosts
alert-monitor - figures out status of various values specified in the *rrds|configfile-rrds* and *hosts|configfile-hosts* config-files
topology-monitor - to analyze changing routes to your monitored hosts

alert-monitor - a status evaluator and alert trigger

Usage:

alert-monitor version 1.27 from remstats 1.0.13b usage: ../alert-monitor [options] where options are:
-d nnn enable debugging output at level ‘nnn’
-f fff use config-dir ‘fff‘[/home/remstats/etc/config]
-h show this help
-G GGG only do hosts in groups ‘GGG’ (a comma-separated list)
-H HHH only do hosts ‘HHH’ (a comma-separated list)

-K KKK only do hosts with keys 'KKK' (a comma-separated list)
-s sss search 'sss' data samples for values [5]
-u generate alerts for hosts unreachable via a down host

Description:

The `alert-monitor` compares the current value of variables specified in the *alerts file|configfile-alerts* in the configuration directory with threshold values and sets the status of those variables accordingly. It saves the current status of variables in `/home/remstats/data/ALERTS`. What value corresponds to what status level is set in the *rrd definition|configfile-rrds* or sometimes the *host definition|configfile-hosts*. This way an rrd definition will specify generally reasonable levels, but they can be overridden for hosts where they aren't reasonable.
For an rrd definition, an alert line looks like:

```
alert varname relation oklevel [warnlevel [errorlevel]]
```

or

```
alert varname nodata status
```

[The latter says that missing data for variable `varname` will cause its status to be level `status`.]
For a host-specified alert level, the line looks like:

```
alert rrdname varname relation oklevel [warnlevel [errorlevel]]
```

or

```
alert rrdname varname nodata status
```

and the interpretation is the same, except that you're having to say which rrd this alert refers to.

The available relations are:

```
< (value is less than threshold)
> (value is greater than threshold)
= (value is equal to threshold)
|< (absolute value of value is less than threshold)
|> (absolute value of value is greater than threshold)
delta< (difference between last two values is less than threshold)
delta> (difference between last two values is greater than threshold)
<daystddev (value is outside threshold * the past day's standard-deviation
<weekstddev (value is outside threshold * the past day's standard-deviatio
<monthstddev (value is outside threshold * the past day's standard-deviati
```

Example

To make things more concrete for the first (normal) case, here's a real example, from the `load` rrd supplied in `config-base`:

```
alert load5 < 3 7 10
```

This means that if the `load5` variable is less than 3, the status is set to OK. If it's less than 7, it's WARN, less than 10 it's ERROR and more than that, it's CRITICAL.
Since the first match is taken, it's possible to leave out the upper levels if you don't want them to ocurr. For example if you only wanted `load5` to ever go to WARN level, never above, you could use:

```
alert load5 < 3
```

and then the only possible status levels are OK and WARN.
The possible `relation`s are: <, =, >, |<, |>, delta<, delta>. The first three should be obvious. The next two allow comparisons to the absolute value of the variable's current value. The last two allow comparisons to the change in value.

Causing alerts

Depending on the lines in the *alerts file|configfile-alerts*, the status may also trigger alerts. A matching line in the *alerts config-file|configfile-alerts* will cause `alert-monitor` to run the *alerter* for each of the specified recipients. It will also be passed, in order:

recipient - the recipient; for *alert-email* it will be an email address
hostname - the name of the host that the alert applies to

ip - the IP number for that host, in case it's not in DNS

rrdname - the name of the RRD

wildpart - the wild part of a wildcard RRD. E.G, for an RRD of `port-ftp` (using the wildcard RRD `port-*`) the wildpart would be `ftp`.

variable - the name of the variable

status - the current status, as decided by alert-monitor

old_status - the previous status

value - the current value of the variable

relation - the relation used to compare the variable to the threshold, mostly for creating informative messages

threshold - the threshold value that was exceeded

start - timestamp of when the alert started

duration - number of seconds that the alert has been active

host-description - the description field from the host config-file

rrd-description - the description tag on this rrd (desc="xxx")

webmaster - the email address of the remstats person

template - the name of the template file to generate the message from.

alerter - construct and send alert text

Usage:

alerter version 1.15 from remstats 1.0.13a usage: alerter [options] args where options are:
-d ddd set debugging output to level 'ddd'
-h show this help
-f arg use 'arg' as config-file [/home/remstats/etc/config]
towho host ip realrrd wildpart var status old_status value relation
threshold alertstart duration hostdesc rrddesc webmaster template

Description:

[`Alerter` may be rolled into the `alert-monitor` at some point in the future. It was easier to test as a separate program, and the performance hasn't been an issue for me.]

`Alerter` is passed its parameters (specified above) by the *alert-monitor*. Most of them are used to fill in information in the text of the alert. The interesting ones are `towho` and `template`.

It also reads the *alert-destination-map config-file|configfile-alert-destination-map* to decide where the alert needs to go. This will give it a list of (method, address) pairs.

For a given template-name, say `xxx`, and method, say `method`, it will look for files in /home/remstats/etc/config/alert-templates, called:

```
method-xxx
method-DEFAULT
xxx
DEFAULT
```

and take the first one it finds. Similarly, it will look for a header to add to the top of the template called:

```
method-HEADER
HEADER
```

and a footer in one of:

```
method-FOOTER
FOOTER
```

The three pieces will be concatenated giving the template text. Then substitutions will be done for the following *magic cookies|cookies*:

```
HOST IP REALRRD WILDPART FIXEDRRD VAR STATUS OLDSTATUS
VALUE RELATION THRESHOLD START DURATION HOSTDESC RRDDESC
NOW TEXTNOW ALERTHOST TOWHO WEBMASTER HTMLURL CGIURL HOSTINDEXURL
```

This gives the alert text. From the method definition in the *alert-destination-map config-file|configfile-alert-destination-map* `alerter` knows which program to run to send the alert text

to the appropriate address, and it does it.

Alert-Sending Scripts

These are now easy to write, and in many cases you won‘t even have to write one. There are two requirements for an alert-sending script:

1) It must take an address to send to on the command-line, and

2) It must accept the text on stdin.

E.G. you could use sendmail with no wrapper.

alert-email - an alert sending script

This is a simple script intended to be run by the *alerter*. Like all alert-senders, it takes one argument on the command-line: the "address" to send the alert to. The text of the alert is fed to this script on stdin.
It sends the alert text to "address" via email, by invoking sendmail, though there‘s no reason that it couldn‘t be re-written to do an SMTP injection directly if some-one wanted to. With no error-checking, it could be re-written as:

```
#!/bin/sh
/usr/lib/sendmail "$1"
```

alert-syslog - an alert sending script

alert-syslog version 1.1 from remstats 1.0.13b usage: alert-syslog [options] facility:severity Where options are:
-d ddd set debugging output to level ‘ddd’
-f fff set config-dir to ‘fff’ [/home/remstats/etc/config]
-h show this help
-l lll use ‘sss’ for logger program
Facility comes from the list:
And severity comes from:
kern user mail daemon auth lpr news uucp cron local0 - local7
emerg alert crit err warning notice info debug
This simply raises an alert by writing it to syslog. Useful if you want to keep everything in syslog.

alert-winpopup - an alert sending script

This is a simple script intended to be run by the *alerter*. Like all alert-senders, it takes one argument on the command-line: the "address" to send the alert to. The text of the alert is fed to this script on stdin.
It sends the alert to use "address", in this case a windows NetBIOS machine name, via a Windows popup message.

alert-yahoo - an alert sending script

This is a simple script intended to be run by the *alerter*. Like all alert-senders, it takes one argument on the command-line: the "address" to send the alert to. The text of the alert is fed to this script on stdin.
It sends the alert to "address", in this case a yahoo ID.

ping-monitor - determine reachability status

Usage:

ping-monitor version 1.10 from remstats 1.0.13b usage: ping-monitor [options] where options are:
-d nnn enable debugging output at level ‘nnn’
-f fff use ‘fff’ for config-dir [/home/remstats/etc/config]
-h show this help
-s sss examine ‘sss’ samples [5]

Description:

The `ping-monitor` looks at the last 5 samples (by default) of ping data and determines the status of the host. It will choose one of the following four statuses:

UP - the host is up now and has always (throughout the sample period) responded to pings.

UPUNSTABLE - the host is up now, but on at least one of the samples, it did not respond.

DOWNUNSTABLE - the host is not responding now, but it has responded within the sample period.

DOWN - the host is down now and has not responded within the sample period.

It also writes coloring information for the ping-index web-page.

topology-monitor

Usage:

topology-monitor version 1.9 from remstats 1.0.13b usage: topology-monitor [options] oldfile newfile where options are:

```
-d          enable debugging output
-f fff use 'fff' for config-dir [/home/remstats/etc/config]
-h          show this help
```

Description:

The `topology-monitor` runs out of *do-traceroutes* to find changes in network paths to the monitored hosts. After `do-traceroutes` has run traceroute for each monitored host, the `topology-monitor` compares the current network path to that host with the previous path. Currently, all that is done with this information is to log when it changes.

Pagemakers

The remstats pagemakers make web-pages, and update other information used in making web-pages. They only need to be run when the configuration file has changed and *run-remstats* is smart enough to do that.

graph-writer - makes web-pages with the graphs and links them together

snmpif-setspeed - sets maximums on snmpif-* rrds

datapage-interfaces - makes datapages for every snmpif-* rrd

datapage-inventory - lists all monitored hosts, uptime, software and hardware

snmpif-description-updater - updates the descriptions for snmpif-* rrds from the SNMP descriptions

graph-writer

Note: `graph-writer` has been dropped in favour of the new *page-writer* which is much more flexible.

snmpif-setspeed

This is a gross hack which modifies all the `snmpif-*` rrds to set the maximum limits on all monitored interfaces. The input and output bps variables have their maximum set to the the maximum for that interface. The various packet counters have their maximums set to the maximum possible packets-per-second assuming minimum-length packets, for that interface speed.
You may not want to run this if you have better knowledge of what real maximums will be encountered for particular interfaces.

datapage-alert-writer

Usage:

datapage-alert-writer version 1.4 from remstats 1.0.13b usage: ../datapage-alert-writer [options] where options are:

```
-d nnn  enable debugging output at level 'nnn'
-h      show this help
```

Description:

This writes a datapage (see *datapage.cgi|datapage-cgi*) which gets linked into the header of each page as the "Alert Index". It gives a quick overview of alert statuses and values for each host. It ought to be self-explanatory.

datapage-interfaces

This makes a *datapage|datapage-cgi* for each host with `snmpif-*` rrds, showing all those interfaces.

datapage-inventory

This makes a single page showing all the monitored hosts. For each host, it shows:

the uptime (from the `uptime` program or SNMP uptime)

the hardware type (from the `uname` program), if available

the software version (from the `uname` program or the SNMP system.sysDescr)

datapage-status

Usage

datapage-status version 1.8 from remstats 1.0.13b usage: datapage-status [options] file ... where options are:

```
-d          enable debugging output
-e    show run-time errors in generated pages
-f fff use 'fff' for config-dir [/home/remstats/etc/config]
-h          show this help
```

Description

The `datapage-status` program creates a datapage (to be interpreted by *datapage.cgi|datapage-cgi*) showing the current values of all variables in all RRDs for that host, in addition to the usual remstats headers.

page-writer

Note: I intend to completely replace *graph-writer* with page-writer as soon as I'm comfortable with it. If you've made changes to `graph-writer`, then you will need to look at `page-writer` to see how to incorporate your changes. Luckily, `page-writer` is intended to be configurable, unlike `graph-writer`, so you oughtn't to need to change the program, only the page templates.

Usage:

page-writer version 1.14 from remstats 1.0.13b usage: ../page-writer [options] collector where options are:

```
 -d nnn  enable debugging output at level 'nnn'
 -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
 -h      show this help
 -p ppp  use 'ppp' as the pages file instead
```

Description:

Like *view-writer*'s templates, it generates pages from templates with special mark-up. It is triggered as a *pagemaker|pagemakers* and attempts to generate the pages specified in the *pages config-file|configfile-pages*.

Simple Tags

The first lot are the most important as they produce the graphs:

<REMSTATS::GRAPH imagedir host rrd graph graphtime>

- which gets translated to a <RRD::GRAPH...> tag for `rrdcgi` to generate the requested graph.

Note: don't get attached to the form of the tag as I'm considering modifying it to something like:

```
<REMSTATS::GRAPH HOST="host" RRD="rrdname" GRAPH="graphname"
[TIME="timename"] [START="starttime"] [END="endtime"] [SIZE="XxY"]>
```

to accomodate the need for extra optional things, like overriding the size.

<REMSTATS::CUSTOMGRAPH imagedir customname graphtime> - which get translated to a <RRD::GRAPH ...> tag for rrdcgi.

The next lot make it easy to produce the original look of remstats:

<REMSTATS::HEADER title> - produces a remstats standard page header

<REMSTATS::STATUSHEADER> (for hosts status headers) and

<REMSTATS::RRDHEADER> (for graph headers) and

<REMSTATS::BAREHEADER> - for page-top HTML up to the begining of the body.

<REMSTATS::TOOLBAR hostname> - produces a remstats host toolbar

<REMSTATS::FOOTER> - produces a remstats page footer

The next group are to allow you to design your own look:

<REMSTATS::INCLUDE filename> - translated to <RRD::INCLUDE ...> for rrdcgi

<REMSTATS::TEMPLATE templatename> - inserts a copy of the named template at this point, substituted as usual. (See <REMSTATS::PAGE...> below.)

<REMSTATS::VAR varname> - inserts the value of the variable "varname" (usually one of: NAME, FILE, GROUP, HOST, RRD, GRAPH or GRAPHTIME)

<REMSTATS::VAR::WILD varname> - inserts the value of the variable "varname" (usually one of: NAME, FILE, GROUP, HOST, RRD, GRAPH or GRAPHTIME), but doesn't substitute wild graph-names.

<REMSTATS::VAR::FIXED varname> - as <REMSTATS::VAR...> but applies the to_filename function to the value so that it can be used in a path (file or URL) without worrying about, e.g., "/" characters screwing things up.

<REMSTATS::VAR::FIXED::WILD varname> - as <REMSTATS::VAR...> but applies the to_filename function to the value so that it can be used in a path (file or URL) without worrying about, e.g., "/" characters screwing things up, but doesn't substitute wild graph-names.

<REMSTATS::SET::VAR varname the value> - define a variable for later use.

<REMSTATS::SET::VAR::FIXED varname the value> - as <REMSTATS::SET::VAR...>, but applies the to_filename function to the value as does <REMSTATS::VAR::FIXED...>.

<REMSTATS::SET::VAR::CONFIG varname key1 ...> - sets a variable to a value from the configuration specified by the keys

Iterator Tags

Thes tags have begin and end markers and replicate the text between them. For example, a <REMSTATS::FOR::GROUP ...> tag will replicate the enclosed text once for each selected group.

<REMSTATS::FOR::GROUP [NAME="groupname"] [LIST="g1,g2,..."]> ... </REMSTATS::FOR::GROUP>

<REMSTATS::FOR::HOST [NAME="hostname"] [LIST="h1,h2,..."] [WITHIN="GROUP"] [TAG="TAG"] ... </REMSTATS::FOR::HOST>

<REMSTATS::FOR::RRD [NAME="rrdname"] [LIST="r1,r2,..."] [WITHIN="HOST"] [TAG="TAG"] ... </REMSTATS::FOR::RRD>

<REMSTATS::FOR::GRAPH [NAME="rrdname"] [LIST="g1,g2,..."] [WITHIN="RRD|CUSTOMGRAPH"] ... </REMSTATS::FOR::GRAPH>

<REMSTATS::FOR::GRAPHTIME [NAME="rrdname"] [LIST="t1,t2,..."] [WITHIN="GRAPH"] ... </REMSTATS::FOR::GRAPHTIME>

<REMSTATS::FOR::CUSTOMGRAPH [NAME="customgraphname"] [LIST="c1,c2,..."] [WITHIN="HOST"] ... </REMSTATS::FOR::CUSTOMGRAPH>

All of these group everything inside them and evaluate the contents with the appropriate variables set as specified. There are four cases:

1 The tag has no parameters. This means that the block will be replicated once for each instance of that kind of thing. E.G. the `<REMSTATS::FOR::HOST>` tag with no parameters will be replicated as many times as there are hosts, and each copy will have HOST set to one of the possible hosts, sorted alphabetically.

2 The tag supplies a variable the same as its type. E.G. `<REMSTATS::FOR::HOST HOST="xxx">` This will produce one copy of the block, with the variable set to the specified value (in the example, HOST will be "xxx").

3 The tag supplies a parameter `LIST="x,y,..."`. This kind of tag will be copied as many times as there are elements of the list (comma-separated, no spaces). The copies will be ordered as they are in the list.

4 The tag supplies a parameter `WITHIN="xxx"`. This will select a subset of the items:

`HOST WITHIN="GROUP"` - gives all the hosts in the current group, as specified in the `GROUP` variable, sorted by host-name.

`RRD WITHIN="HOST"` - gives all the rrds that the current `HOST` has on it, in the order they were specified in the *hosts config-file|configfile-hosts*.

`GRAPH WITHIN="RRD"` - gives all the graphs defined for the current `RRD`, in the order they were defined. If `HOST` is defined, then it will obey `nograph` directives from the hosts config-file.

`GRAPH WITHIN="CUSTOMGRAPH"` - gives all the graphs defined on the current `CUSTOMGRAPH`, in the order they were defined.

`GRAPHTIME WITHIN="GRAPH"` - gives all the graphtimes specified on the current `GRAPH`, in the order they were specified.

`GRAPHTIME WITHIN="CUSTOMGRAPH"` - similarly for customgraphs.

`CUSTOMGRAPH WITHIN="HOST"` - gives all the customgraphs specified for the current `HOST`, in the order they were specified.

The <REMSTATS::PAGE ...> Tag

The powerful one which allows a page to cause the generation of other pages.

<REMSTATS::PAGE pagename filename>

This will cause page-writer to generate the specified page at this point. If it's within one of the iterators, then it will be triggered multiple times with various values for the variables. This way, specifying a single page in the *pages config-file|configfile-pages*, can trigger the generation of a whole family of pages.

The <REMSTATS::CHDIR ...> Tag

This changes the current directory for page generation. This allows a PAGE directive to generate a full sub-directory tree of pages, with the correct templates. Note that **it will create missing directories, but not missing hierarchies**. This is intentional; it would be tedious and error-prone to have to create them by hand. It will, however, note as a non-fatal error when it creates directories.

The <REMSTATS::CHMOD mode file> Tag

Changes the mode (an octal number) of the `file`. Mainly to make the GRAPHS sub-directories writable by the apache group.

The <REMSTATS::CHGRP group file> Tag

Changes the group of the file `file` to `group`. Mainly to make the GRAPHS sub-directories writable by the apache group.

The <REMSTATS::MKDIR ...> Tag

This is identical to the <REMSTATS::CHDIR> tag, except that it doesn't change the current directory. Note that it won't complain if the directory already exists.

Variable substitution within tags

Since I don't want to get into the hairy parsing required, you can't embed tags within other tags. I've found, so far, that the only thing that I've needed to insert into a tag was the value of a variable, so that's the only non-simple-text that you can use inside a tag.

To insert the value of a variable, include a string like "##XXX##" where "XXX" is the name of the variable whose value you want.

Avoiding extra whitespace

End a line in a page template with a backslash "\" and the newline that would normally be present will not be passed on to the generated page.

view-writer

Note: The `view-writer` is obsolete and replaced by the new *page-writer*. It is capable of doing everything `view-writer` can, and much more. If there's anything you're using of `view-writer` that you think `page-writer` doesn't do, please let me know. If it still works for you, fine, but I'm not interested in it any more.

Description:

The `view-writer` makes web-pages from the view definitions which are contained in the *views config-dir|configfile-views*. All the documentation relating to `view-writer` is there too, as it only implements what the views request.

run-remstats2

Usage:

run-remstats2 version 1.10 from remstats 1.0.13b usage: ../run-remstats2 [options] [—options-to-be-passed] where options are:

—debug=nnn enable debugging output at level 'nnn'

—config_dir=fff use 'fff' for config-dir [/home/remstats/etc/config]

—help show this help

Description:

[`Run-remstats2` is a replacement for *run-remstats*. If you're converting to use `run-remstats2`, then you need to know that the old `collectors`, `monitors` and `pagemakers` lines in the *general config-file|configfile-general* are ignored by `run-remstats2`.]

`Run-remstats2` is the main process part of remstats. It looks at the *run config-file|configfile-run* to get a list of the stages it is to go through, and their order. These stages are defined in the *run-stages config-dir|configfile-run-stages*. Each stage consists of one or more named command-lines (from a single program, to a pipeline with many options. All the requested command-lines for a stage are run in parallel. `Run-remstats2` (like `run-remstats`) will then wait for all the commands to finish, before proceeding with the next stage. There are more details to controlling this process, but they are documented with the `run` and `run-stages` documentation.

Each command will have stderr redirected to a separate file, and if the command is a pipeline, each part of the pipeline will have stderr directed to it's own file, thus keeping each program's error output separate. These files are stores in the `tmp/run-stages` directory, with names like `[stage]-[id]-out-[pid].[pipeline-part]`.

Notes

When starting, it will look for a file in the tmp directory called `STOP-run-remstats2` and if it exists, will exit immediately.

run-remstats - run a complete cycle

Usage:

run-remstats version 1.20 from remstats 1.0.13b usage: ../run-remstats [options] [—options-to-be-passed] where options are:

—debug=nnn enable debugging output at level 'nnn'
—config_dir=fff use 'fff' for config-dir [/home/remstats/etc/config]
—locking create a lock-file to prevent two instances running
—help show this help

Description:

This is obsolete. You should be running *run-remstats2* instead.

`run-remstats` is the main script for a remstats collection machine. As a simplified overview:

check-config is run first.

In parallel, all the *collectors* are run, each feeding it's own *updater* process. Some of them query remstats *servers*, some get their information in other ways. So you have a bunch of pipelines like:

```
xxx-collector | updater xxx
```

When all the collectors have finished, the *monitors* get run in parallel to figure out what's happening.

Afterwards, if the configuration directory has changed, run the *pagemakers*, to re-do the web-pages.

Finally, it prints all the stderr output of all the various programs, separated by program.

For each of these programs, `run-remstats` will set a timer (see `watchdogtimer` in the *general config-file|configfile-general*). If the timer expires and the program is still running, `run-remstats` will kill that process. This avoids the problem of a hanging collector hanging the whole remstats cycle.

It also manages a lock-file to make sure that two instances don't run concurrently. The lock-file's name is based on the name of the `run-remstats` script. (See **Running multiple copies of run-remstats** below.)

It keeps a status file in the configured temp directory (`/home/remstats/tmp` by default) which is used by *monitor* to show where the `run-remstats` process has gotten to.

When starting, it will also look for a file in the tmp directory called `STOP-run-remstats` (default), and if it exists, will refuse to run at all.

Running multiple copies of run-remstats

If you symlink `run-remstats` to `run-remstats-XXX`, then the default configuration directory for `run-remstats-XXX` will be `/home/remstats/etc/config-XXX`. Since the lock-file is named for the script which invokes it, you won't have collisions between the two instances, as long as your configuration files don't conflict. You can have multiple collector-only instances collecting data which is formatted by a single pagemaker instance, (in theory) but this will require at least three config-dirs which must be closely co-ordinated. If you want to do this for performance reasons, I do plan to address this in future.

Configuration:

See the *general|configfile-general* config-file. The lines to configure run-remstats are:

```
pinger, collectors, monitors, pagemakers, watchdogtimer
```

check-config

Usage:

check-config version 1.25 from remstats 1.0.13b usage: check-config [options] where options are:

```
-c      write SNMP communities
-C CCC  set configuration-debugging output to level 'CCC'
-d ddd  enable debugging output at level 'ddd'
-D      dump configuration (must have DEBUG enabled in fixup.config)
-e      print environment variables to stdout
-f fff  use config-dir 'fff' [/home/remstats/etc/config]
-h      show this help
-l lll          list 'lll' on stdout (comma-separated list of: host, ip, rrd)
-s sss  shell type to use for -e [sh]
-t      test-mode; do not make any changes
```

Description:

`check-config` uses the common `read_config` routine to read the configuration file which will confirm that it can be read successfully by other programs. It also makes sure that the data directories for all hosts exist, and creates new RRDs.

run-remstats also uses the `-e` option to make basic configuration information avaiable to the shell.

remstats-monitor - watch remstats processes

Usage:

```
remstats-monitor [sleeptime]
```

where sleeptime is the time (in seconds) to wait between polls

Description:

This is primarily a development tool. It loops doing a `ps` command, weeding out everything except the remstats processes and cleaning up the results, to make it easier to read. It also shows the process-id from *run-remstats* lock-file, and the status from its status file.

It's not written portably and will probably have to be tweaked by hand if you want to run it. If you find it of interest, please let *me|mailto:terskine@users.sourceforge.net* know.

CGI Scripts

These are intended to be invoked via the html-writer created toolbars, to do the supplied functions to the host in question.

alert.cgi|alert-cgi - Shows the current alert status of selected rrd variables.

availability-report.cgi|availability-report-cgi - Shows availability of RRD variables.

dataimage.cgi|dataimage-cgi - Generates images based on live data.

datapage.cgi|datapage-cgi - Generates web-pages containing dynamic data.

graph.cgi|graph-cgi - Allows non-remstats web-pages to show remstats graphs.

log-event.cgi|log-event-cgi - log a manual event.

ping.cgi|ping-cgi - Ping the host.

showlog.cgi|showlog-cgi - Display selected portions of the remstats log files.

traceroute.cgi|traceroute-cgi - find network path to a host

whois.cgi|whois-cgi - look up information about hosts, IP#s, ...

Note: Access to the various CGIs is controlled by the *access config-file|configfile-access*.

alert.cgi - Alert Reporting and Updating

This CGI script will generate the Alert Report and also let you modify it. You can turn alerts off for a specific line (by checking the `quench` check-box). You can also attach comments to a specific alert, for example to let other people know that you're already working on it, or when a service will be available again.

availability-report.cgi

The `availability-report.cgi|availability-report-cgi` calls *availability-report* to produce a report of "availability" according to the definitions in the *availability|configfile-availability* config-file.

dataimage.cgi - create images driven by live data

Usage:

```
<IMG SRC="/remstats/dataimage.cgi?imagename">
```

Description:

`Dataimage.cgi` reads an image definition from `/home/remstats/datapage/imagename.image`. Some of the commands are in common with *datapage.cgi|datapage-cgi.html#common_commands* and are documented there:

```
oid, rrd, status, eval, debug, macro, macroend and *EOD*
```

These retrieve and manipulate data. There are also commands to create images:

```
image, colordef, color, linewidth, line, rectangle, circle,
fill, font, text, out, flow
```

Image Commands

image

The image command has two formats. The first looks like:

```
image WIDTH HEIGHT
```

This creates a blank image of the size specified. Sometimes you'll want a background for the image, and you can use the second form to specify a file to read for the background:

```
image BGFILE
```

This will create the new image the same size as the one in `BGFILE`, by reading `BGFILE` and using its contents as the background. N.B., the image must be a PNG graphic.
The `image` command also defines a few colors (see `colordef` below): `black`, `white` and `transparent`, sets the current color to `black`, fills the image with `white` and sets the `linewidth` to 1.

colordef

[It can also be spelled `colourdef`.]
This defines a new colour and names it. The command looks like:

```
colordef COLORNAME RED GREEN BLUE
```

where `RED`, `GREEN` and `BLUE` specify the level of each of those colours to be mixed to define the colour referred to in the script as `COLORNAME`.

color

[It can also be spelled `colour`.]
This sets the current colour, to be used by those commands that don't specify a colour. It is used as simply:

```
color COLORNAME
```

linewidth

This sets the width of lines. It isn't honoured by all other commands, unfortunately, but so far this hasn't been a problem for me. It looks like:

linewidth WIDTH

line

This just draws a line in the current `color` and `linewidth`:

```
line X1 Y1 X2 Y2
```

rectangle

This is a way to draw a rectangle, without useing `line` 4 times:

```
rectangle X1 Y1 X2 Y2 [filled]
```

The co-ordinates (`X1`, `Y1`) and (`X2`, `Y2`) define oposite corners of the rectangle. If the keyword `filled` is added to the end, the rectangle will be filled with the current colour as well.

circle

Here you get a circle:

```
circle X Y RADIUS [filled]
```

The circle will be centered on (`X`, `Y`) with a radius of `RADIUS`. If the keyword `filled` is added to the end, the circle will be filled with the current colour as well.

fill

This command permits you to fill arbitrary regions:

```
fill X Y [COLORNAME]
```

The `COLORNAME` is optional.

text

The `text` command sets text into the image, for labelling things:

```
text X Y TEXT
```

font

This changes the font for the `text` command:

```
font (giant|large|mediumbold|medium|small|tiny)
```

out

This permits the script to output additional information to an auxiliary file. I added this for doing image-maps automatically, which can be automatically loaded by a *datapage.cgi|datapage-cgi* web-page.
The syntax is:

```
out TEXT
```

flow

This draws a strange double-headed, bi-coloured arrow. Think of it as two half arrows, split lengthwise, one in each direction. The colour and width of each half arrow indicates the flow in that direction. I use it for indicating network traffic flow, which usually isn't the same in both directions. It looks like:

```
flow X1 Y1 X2 Y2 INFLOW OUTFLOW
```

The co-ordinates (`X1`, `Y1`), (`X2`, `Y2`) indicate the ends of the flow. `INFLOW` and `OUTFLOW` indicate the level in each direction, relative to (`X1`, `Y1`).

datapage.cgi - dynamic data in web-pages

Usage:

```
<A HREF="/remstats/datapage.cgi?pagename">whatever</A>
```

Data Collection

`Datapage.cgi` looks for the page definition in the file `/home/remstats/data/DATAPAGES/pagename.page` The page definition is in two parts, separated by a line like:

```
BEGIN-PAGE
```

The first part's purpose is to define variables to be included in the second part, which is an HTML template, with magic tags.

All lines in the first or definition part are subject to variable interpolation. Any ocurrance of `${variablename}` will be replaced by the current contents of the variable `variablename`. This will be done up to five levels, permitting expansion of ${${h}_${interface}}, providing that you've got values for the variables `h` and `interface`. N.B. variable names must be lower case.
In addition, within a macro expansion, macro-arguments will also be interpolated, before variable interpolation, for all ocurrances of `${ARGNAME}`, assuming that there is an argument for the current macro called `ARGNAME`. N.B. macro argument names must be UPPER case.

Common Commands

The commands permitted in the first part are:

```
oid, rrd, status, eval, debug, macro, macroend,
alertstatus, alertvalue and *EOD*
```

These commands are in common with the *dataimage.cgi|dataimage-cgi* script, but are only documented here.

oid

This fetches an SNMP value into a datapage variable. The command looks like:

```
oid VARNAME HOSTNAME OIDNAME
```

The `VARNAME` is the name of the datapage variable (let's just call them variables from now on).
The `HOSTNAME` is the name of the host to query. The SNMP community-string is usually supplied in that host's config-file, but can be supplied in usual MRTG fashion by giving `COMMUNITY@HOSTNAME` or even `COMMUNITY@HOSTNAME:PORTNUMBER` instead of the `HOSTNAME`.
The `OIDNAME` must be defined in the *oids config-file|configfile-oids*, but can be suffixed by the usual numbers. E.G. you can use ifName.4 to get the ifName for interface 4.

rrd

This fetches a value from an RRD database into a variable. It looks like:

```
rrd VARNAME HOSTNAME RRDNAME DSNAME CF
```

The `RRDNAME` is the name of the rrd, as remstats knows it, not fully qualified. I.E. it will be under the config-file defined `datadir`, and under the host's directory under that.
The `DSNAME` is the ds-name within that RRD file and the `CF` is the usual RRD consolidation-function to be applied.

status

This is so-named because it fetches remstats status files, usually written by the various *collectors* and *monitors*. It looks like:

```
status VARNAME HOSTNAME STATUSNAME
```

The `STATUSNAME` is the name of the status file, as named in the host's data directory. There is a standard mapping applied by the function `to_filename` from the `remstats.pl</t> file to munge the filename so that it won't conflict with the filesystem. Either look for the name in the data directory, use the function (see eval) or look at the code. I` **`am`** `planning on changing the mapping when I figure out the best way to do it.`

eval

The eval command lets you modify the values fetched by previous `oid`, `rrd`, `status` and `eval` commands with arbitrary perl code. It looks like:

```
eval VARNAME PERLEXPRESSION
```

The `PERLEXPRESSION` is a perl expression and can be arbitrarily complex, but gets messy quickly with the `datapage.cgi` and perl both doing variable interpolation.
Note: `datapage.cgi` uses *private.pl|private*, so you can include commonly used functions here to make your datapage creation easier.

debug

The `debug` command takes a number which is the level to set debugging to. It causes extra output which may be helpful in figuring out why your page isn't working the way you expected.

alertstatus

This lets you fetch the alert level for a given (host, rrd, dsname, cf) combination. The command looks like:

```
alertstatus VARNAME HOSTNAME RRDNAME DSNAME [CF]
```

This will fetch the alert status and put it in the datapage variable VARNAME. The status will be the same set of values shown on the *alerts report|alerts-cgi* for status. The CF parameter is optional and is rrdtool's consolidation function. It will be set to AVERAGE if it's not supplied.

alertvalue

This is the same as `alertstatus` except that it sets the variable to the current value of the (host, rrd, variable, cf) combination.

The HTML template

This is almost just HTML with a few magic tags inserted. The difference is that the beginning must include HTTP headers. If you don't want anything fancy, just begin like:

```
------ cut here ------
BEGIN-PAGE
content-type: text/html

------ cut here ------
```

Note: the empty line after `content-type:` is **not** optional. It's necessary to end the HTTP headers. The magic tags are:

<DATAPAGE::STATUS host statusfile>
: inserts a specified status file

<DATAPAGE::VAR varname>
: interpolates the value of a datapage variable

<DATAPAGE::HEADER title>
: generates a standard remstats header

<DATAPAGE::STATUSHEADER hostname>
: generates the status headers for the named host

<DATAPAGE::TOOLBAR hostname>
: generates the toolbar for the named host

<DATAPAGE::FOOTER>
: generates a standard remstats footer

<DATAPAGE::INCLUDE filename>
: include the contents of a file from the datapage directory, for imagemaps ...

<DATAPAGE::PATHINCLUDE filename-with-path>
: include contents of a file specified with a complete path

<DATAPAGE::MACRO macroname [argvalue] ...>
: include boilerplate HTML with substitutions

<DATAPAGE::GRAPH host rrd graph time>
: generate the specified remstats graph

<DATAPAGE::CUSTOMGRAPH graph time>
: generate the specified remstats customgraph

<DATAPAGE::ERRORS>
: inserts the text of errors encountered in generating the page. Without this one, you won't see any errors. That way you include the errors and debugging output (see next item), which you're creating/debugging the datapage and afterwards turn them off. The errors and debugging output may include information you don't want to reveal to outsiders. Also, collecting all the error output together avoids spoiling the formatting of the page.

<DATAPAGE::DEBUG>
: inserts debugging output. Without it, you won't see any debugging output.

graph.cgi - exporting remstats graphs

The purpose of `graph.cgi` is to allow remstats graphs to appear on external (not part of remstats) web-pages. It's **not** efficient, with the graphs being generated whenever the page is reloaded, but it is portable. All you do is to create an <IMG SRC...> tag with the appropriate values, like:

```
<IMG SRC="/remstats/graph.cgi?host=aaa&rrd=bbb&graph=ccc&graphtime=ddd">
```

and replace `aaa` with the name of the host, as remstats knows it, `bbb` with the name of the RRD, `ccc` with the name of the graph within that RRD, and `ddd` with the name of the timespan, from the *times config-file|configfile-times*. If the RRD is a wildcard RRD, e.g. `snmpif-*`, then you must use the specific instance, e.g. `snmpif-eth0`.
The `graphtime` specifier may be replaced by `start=xxx` and `end=yyy` or omitted altogether. If `start` and `end` are specified, the permitted formats are documented in the manpage for rrdfetch. If you use the form:

```
hh:mm dd.mm.yyy OFFSET
```

with OFFSET being a string composed of multiple ocurrences of plus or minus, a number and the units (s|min|h|d|mon|y). E.G. +2h+40min or -20d+5h
The graph's defined size may also be overridden using `width=www` and `height=hhh`. These specify the size, in pixels, of the graph portion of the image, not the whole image, as usual with rrdtool.
That's all there is to it.

log-event.cgi - log events from a web-page

This shows up on the *showlog.cgi|showlog-cgi* as a link to permit you to manually enter events into the log. Don't feel obliged to enter all the fields. The data isn't checked for meaning, just for syntax. In other words, a host-name must look like a host-name, but it doesn't have to be a real host-name.
This cgi-script ought to be protected. See the *web-server installation|install-webserver* docs.

macinfo.cgi

Usage:

In a URL like:

```
<A HREF="/remstatsmacinfo.cgi?host=10.11.12.13">macinfo for 10.11.12.13</A
```

Description:

It simply re-formats the output of *macinfo* into HTML and wraps remstats headers and footers around it.

ping.cgi

The `ping.cgi` script allows you to ping a host. It's intended to be called off a host's toolbar, but that's not required. Simply provide the hostname or IP number and it'll ping it.
As an example, you could ping ftp.uu.net with a URL like:

```
L</remstats/ping.cgi?host=ftp.uu.net|/remstats/ping.cgi?host=ftp.uu.net>
```

show-config.cgi

Usage:

```
&lt;A HREF="/remstats/cgi-bin/show-config.cgi?xxx=yyy"&gt;show-config&lt;/A

<A HREF="/remstats/cgi-bin/show-config.cgi?xxx=yyy">show-config</A>
```

where `xxx` is one of host, rrd, tool, time, archive and `yyy` is the name of one of whatever `xxx` refers to. E.G. of `xxx` is `host`, then `yyy` must be the name of one of the defined hosts.
The result has links to `show-config.cgi|show-config-cgi` for other kinds of configuration entry where it makes sense and links to the documentation where that makes sense.
Try it.

showlog.cgi

Any alert is also logged to the remstats log files (one file per day). Other information is also logged, for example, the *topology-monitor* logs network topology changes. The `showlog.cgi` script allows you to display selected portions of the log files, by time-period, by host, ...

traceroute.cgi

This script uses *traceroute* to find the path from the remstats host to some other specified host. It's intended to be called off a host's toolbar, but that's not required. For example, you could trace the path from here (trevelyan.sourceworks.com) to ftp.uu.net with a URL like:

```
L</remstats/traceroute.cgi?host=ftp.uu.net|/traceroute.cgi?host=ftp.uu.net
```

There are other options that you can specify:

`no_names` - just shows IP numbers instead of looking up the domain-names

`ASNs` - look up the Autonomous System Numbers (ASNs) for the IP number of each hop. It can be useful for figuring out which networks you are traversing.

`owners` - look up the "owner" via SOA records

`fast` - continues on to the next hop as soon as the current one answers

whois.cgi

This script talks to the ARIN whois database (by default) to look up network names, IP numbers and AS numbers. It's usually linked into the results of *traceroute.cgi|traceroute-cgi* so that you can look up what your traceroute results actually mean.
Try `traceroute.cgi` if you want to see how it works.

Troubleshooting

[Almost all of the various programs accept a `-h` or `–help` flag to ask for help, if you've forgotten or don't know it. Most also accept a `-d` or `–debug` flag to set the level of debugging output.]

Tools

There are two tools which are specifically intended for finding problems:

check-config - You should **always** run this after making any configuration change. It won't find all possible problems with the configuration files, but if it doesn't like your configuration, you can be sure that nothing else will, If it prints nothing, all is well. It will also note when it is creating new RRDs and directories, but this is not an error.

check-rrdlast - checks the last-update time on all RRD files. Make sure that you have no `TIMEWARP` marked RRDs. These are RRDs with a last-update time in the future. They won't be updated until after what they think is the last-update time. Anything that's `STALE` marked is also a concern if it's long overdue. These would be RRDs which aren't getting updated, for some reason.

Are things running?

The first thing to check is whether *run-remstats2* is being started by cron. Look in `tmp` for a file called `LOG-run-remstats2`. If it's there and doesn't have anything marked ERROR or ABORT, this is a good thing. Check that the modified time for the file is within the last 5 minutes, to make sure that it's being run at appropriate times.
The next thing to check is whether *run-remstats2* or *run-remstats* has been told to run the *collector|collectors* at all.

For *run-remstats*, look at the `collectors` line in the `general` config-fileebet

For *run-remstats2*, look at the `run-stages/collectors` file and make sure that the appropriate line is uncommented.
Once it's running, you can check what data it found in the last run. Look in data/LAST/<collectorname>. The lines are all the data which that collector found on the last run.

Remstats Services

First of all, make sure that the perl specified in the shebang line exists and is the right perl to use. If you're getting a message like "`No such file or directory`" and you can see that the script exists, check the first line of the script, which will look something like:

```
#!/usr/bin/perl -w
```

And make sure that the specified perl (in the example "`/usr/bin/perl`" exists, is executable by the user specified in the inetd configuration, and is the correct perl to use.

If you're having difficulty getting data from remstats *servers*, you should check them out without the collectors. For the *unix-status-server*, you can make sure that you have access, by running the following on the server host:

```
% telnet localhost 1957
UNAME
GO
```

If you don't get connected, you'll have to look at your inetd configuration, and the output it logs. You might have to run inetd with debugging flags to get it to log enough to be usefull.
If you get connected and then shortly afterwards disconnected, this is the signature of the tcp_wrappers refusing a connection: check out `/etc/hosts.allow`. You ought to have lines for the remstats servers which you are running on that host:

```
unix-status-server : collector-host localhost
log-server : collector-host localhost
```

Note that inetd will refer to `unix-status` because that's the name of the service from `/etc/services` (I had to shorten the name to `unix-status` from `unix-status-server`, as I ran into a length limit on service names somewhere.) However, `hosts.allow` refers to program names.

Debugging Output

Almost all of the programs will take a `-d ddd` flag to set the debugging output level. Unless you're willing to look at the code, you probably won't want more than level 1, but it can be helpful.
If you're trying to figure out what's happening, it's helpful to run *run-remstats2* interactively with `-d 1`, after disabling it from crontab. This will tell you which processes are being run and how long they're taking.
Follow that by running a collector interactively with the `-d 1` flag, and possibly a `-H xxx` flag to restrict it to host `xxx`.
It's also good to know what the output of a collector looks like. You can get a general description under *collectors*, or you can just run a collector interactively and see what it comes up with. Remember that collectors won't even try to get information that they don't need yet, so if you want to see everything that the collector would get in a normal run, you might want to use the `-F` flag to force collection and/or `-u` to attempt even hosts which are down.

Usefull Files

There are some useful files to look at:

`data/ALERTS` contains all the alert statuses for all variables, whether they have an alert defined or not. It is maintained by the *alert-monitor*.

`data/IP_CACHE` contains IP numbers which were looked up by the *ping-collector* during the current run of *run-remstats2*. This is to save extra DNS lookups for large sites. If you've re-numbered any hosts, you might want to look here to see what remstats was using for an IP number for a host.

`data/LOGS` is a directory containing log-files, one per day. Currently, they are kept for a bit less than a year. You can grep them to do searching in them that *showlog.cgi|showlog-cgi* won't let you do. These show all events which remstats thought might be interesting.

`data/NT` is a directory with information about Windows NT hosts/domains collected by *nt-discover*.

`data/TRACEROUTES` contains all the aggregate traceroute information collected by *do-traceroutes*.

`data/LAST` has the data the last run of the various collectors collected, in files named after the collector. Check here to make sure that your data is being collected, if the graphs are getting no new data. Note that if you have RRDs with different step times, it is normal that the longer step times won't always show up.

under `tmp` are several interesting things. You'll find a file called `LOG-run-remstats2` (or `LOG-run-remstats` if you're still running it instead). This file tells which processes were run during the last run of *run-remstats2* (or *run-remstats*). There is also a file called `LOG-run-remstats2.old` which has the same stuff for the run before.

The file `STATUS-run-remstats2` contains info on the progress of a running instance of `run-remstats2`.

Under `tmp/run-stages` are the files containing the stderr output from the processes started by `run-remstats2`. In an ideal world, you'll only see files in here while `run-remstats2` is running, but if it dies or you kill it, you may have files left in here which may give you an idea of what went wrong before it died.

the pseudo-host `_remstats_` maintains information about the functioning of remstats itself, currently, it's mostly about the performance of the various collectors.

Trouble reporting

Send your report to the mailing-list <remstats-list@lists.sourceforge.net>.

Always include the remstats version. You can find it in the `VERSION` file (just the version number) or the `release` file (status, number and date). Unless you've got a pre-release version, the `VERSION` file has what you need.

Run *check-config* interactively and include the output.

Describe what you ran to produce the error, including command-line args.

Include any error messages you received from cron, or found in any of the above files, even if you don't understand them.

Complain, in detail, about documentation or error messages which you don't understand. They're intended to be comprehensible.

All remstats error messages should be prefixed by `ERROR:` or `ABORT:`. (The only difference is that an abort is an error the remstats can't or shouldn't continue from.) Nothing should trigger any perl errors or warnings; complain about them.

check-rrdlast

Usage:

check-rrdlast version 1.3 from remstats 1.0.13b usage: ../check-rrdlast [options] where options are:

-d nnn enable debugging output at level 'nnn'

-D show the data for the last update

-f fff use 'fff' for config-dir [/home/remstats/etc/config]

Note: the output is one line for each rrd showing: host, rrd, last-update, a pretty-printed version of last-update, (now - last-update), step, and status(OK is ok, STALE means that it is overdue for an update and TIMEWARP means that the last-update time is in the future. The data will be on its own line like:

timestamp dsname=value ...

Description: Shows the last update time for all RRDs and notes if it is in the future or more than the RRD's step time in the past.

do-traceroutes - find the path to each host

This runs traceroute against each host being monitored. After they've all finished, it runs the *topology-monitor*.

I'm planning to make graphical representations of how you are connected to the hosts you're monitoring, but that's not working yet.

traceroute

Usage:

/bin/sh: line 1: ../traceroute: is a directory

Description:

Hmm. I think that describes its use pretty well. What does it do? Oh. Well it sends UDP packets with the time-to-live set to 1, then 2 then 3 and so on. This causes the routers that these packets are sent through to complain after the requisite number of hops. I.E. the first router complains about the first packets, with TTL set to one, the second about the packets with TTL set to two etc. Traceroute catches the complaints and times how long it took. This not only shows you how your packets are getting to the destination, but sometimes, where the congestion is as well. There's a lots better explanation in the source, so if you want more, *UTSL|http://www.tuxedo.org/~esr/jargon/html/entry/UTSL.html*.

This version of traceroute is used in *traceroute.cgi|traceroute-cgi*, which isn't required, just handy on occasion, and in *do-traceroute*, which you don't need unless you're curious about your routing and how

it's changing over time. The only extra options that do-traceroute uses are the `-A` option to look up the ASN (Autonomous System Number) and the `-O` option to look up the DNS owner.

Miscellaneous Scripts

These are scripts that don't really fit in anywhere.

alerter sends an alert
availability-report shows availability of RRD variables
genindex makes an index
genmenu makes the vertical menu-bars used in these docs.
htmlpod makes pod files from html files (roughly).
lockfile makes lock-files
podhtml makes html files from pod files.
podlatex makes LaTeX files from pod files.
podpdf makes PDF files from pod files.
remstast-backup makes a backup of your data and configuration.
rrd-report produces reports from a raw rrd.

These are release-related scripts:

convert-config-links - copies links to files (just read it)

alerter - construct and send alert text

Usage:

alerter version 1.15 from remstats 1.0.13a usage: alerter [options] args where options are:
  -d ddd set debugging output to level 'ddd'
  -h   show this help
  -f fff use 'fff' as configuration directory [/home/remstats/etc/config]
The args are documented in alert-monitor, a quick list:
  towho host ip realrrd wildpart var status old_status value relation
  threshold alertstart duration hostdesc rrddesc webmaster template

Description:

[`Alerter` may be rolled into the `alert-monitor` at some point in the future. It was easier to test as a separate program, and the performance hasn't been an issue for me.]

`Alerter` is passed its parameters (specified above) by the *alert-monitor*. Most of them are used to fill in information in the text of the alert. The interesting ones are `towho` and `template`.

It also reads the *alert-destination-map config-file|configfile-alert-destination-map* to decide where the alert needs to go. This will give it a list of (method, address) pairs.

For a given template-name, say `xxx`, and method, say `method`, it will look for files in /home/remstats/etc/config/alert-templates, called:

```
method-xxx
method-DEFAULT
xxx
DEFAULT
```

and take the first one it finds. Similarly, it will look for a header to add to the top of the template called:

```
method-HEADER
HEADER
```

and a footer in one of:

```
method-FOOTER
FOOTER
```

The three pieces will be concatenated giving the template text. Then substitutions will be done for the following *magic cookies|cookies*:

```
HOST IP REALRRD WILDPART FIXEDRRD VAR STATUS OLDSTATUS
VALUE RELATION THRESHOLD START DURATION HOSTDESC RRDDESC
NOW TEXTNOW ALERTHOST TOWHO WEBMASTER HTMLURL CGIURL HOSTINDEXURL
```

This gives the alert text. From the method definition in the *alert-destination-map config-file|configfile-alert-destination-map* `alerter` knows which program to run to send the alert text to the appropriate address, and it does it.

Alert-Sending Scripts

These are now easy to write, and in many cases you won't even have to write one. There are two requirements for an alert-sending script:

1) It must take an address to send to on the command-line, and

2) It must accept the text on stdin.

E.G. you could use sendmail with no wrapper.

availability-report

Usage:

availability-report version 1.28 from remstats 1.0.13a usage: availability-report [options] where options are:

```
-c      use colors in the output
-d ddd  set debugging output to level 'ddd'
-f fff  set config-dir to 'fff' [/home/remstats/etc/config]
-h      show this help
-H HHH  show only hosts HHH (comma-separated list) [all]
-G GGG  show only groups GGG (comma-separated list) [all]
-R RRR  show only rrds RRR (comma-separated list) [all]
-g      show group summary
-t ttt  availability for time-period ttt (start,finish)
```

Description:

This is mainly intended to be called from *availability-report.cgi|availability-report-cgi*. It provided a report on "availability" of specified RRD variables, by default, all that have definitions in the *availability|configfile-availability* config-file. Exactly what it means for a variable to be "available" is up to you. It's intended to give some measure of when a host or service isn't useable, so, e.g. the default definition of availability for the `ping` RRD variable `rcvd` (number of ping responses received) is:

```
ping rcvd MINIMUM > 0
```

In english, the `rcvd` variable is considered unavailable if:

```
- it is less than or equal to zero (I.E. it didn't respond to ping)
- there is no data available for that time period
```

N.B.: The interaction between rrd archive consolidation and the xff value, (see rrdcreate), can result in longer periods of unavailability or conversely, masking periods of unavailability. Choose the consolidation function carefully to make sure you're getting the best data possible.

convert-config-links

Usage:

usage: ../convert-config-links [-h]

Description:

The problem is that an upgrade installation of remstats will overwrite the config-base directory, but previous installations of remstats created new configuration directories as symlinks to config-base. Some of these files need to be changed and some are commonly changed, specificly:

```
alerts alert-destination-map general html links tools
```

Installing a new version of remstats will overwrite config-base, including these files.

`convert-config-links` is a conversion tool for upgrading from remstats versions before 1.0A. It will convert the commonly changed config-files from symlinks to copies of the appropriate files from config-base. In remstats versions after 1.0A the *new-config* program will "Do the Right Thing" (TM) and make copies by itself, so you'll only have to run this once.

If you're installing remstats for the first time, you can ignore this program.

Remstats Files

This is a reference to the various kinds of files remstats uses and creates.
The simple ones really don‘t need explanation, and are included here for completeness:

`/home/remstats/bin` - contains executables provided with remstats

`/home/remstats/lib` - contains "libraries" required at run-time by the executables. So far, it‘s only `.pl` files.

`/home/remstats/etc` - contains configuration information. Immediately after a new installation, it will only contain a single sub-directory - `config-base`. *new-config* will create configuration directories under `/home/remstats/etc` containing a mix of symlinks to files and sub-directories of `config-base` and copies of those files and sub-directories, according to whether an installation is likely to want to change them.

`/home/remstats/html` - (**FIXME** this will have to be re-written for page-maker) contains the created web-pages to display status and graph information. At the top level, there are the index pages and a sub-directory for each host, named with that host‘s name. The host sub-directories contain all the index pages and graph images for that host. In addition, there are a few special files and sub-directories of `/home/remstats/html`:

- `MOTD.html` - The *graph-writer* will create this file if it doesn‘t exist, but no remstats programs will place anything there. It‘s purpose it to contain site-specific information of importance, similar to the unix `/etc/motd` file.
- `IMAGES` - This contains remstats-supplied static images for various things, mostly logos.
- `CUSTOM` - This subdirectory contains the web-pages and images associated with *customgraphs|configfile-customgraphs*, graphs of information which don‘t relate to a single host. There is a sub-directory for each customgraph, to hold its index pages and images and a top-level index page.
- `VIEWS` - This directory contains view information created by the *view-writer*. There will be an overall index page and a sub-directory for each view to conain its index page and images.
- `MOVIES` - This directory is used by *collect-movie-image* to take snapshots of the netflow *dataimage|dataimage-cgi*, to be assembled into a "movie". There will be files there called `snap-DDHHMM.png`. This will not be usefull to you unless you write your own netflow datapage.

`backup` - This directory is used by *remstats-backup* to contain two tar-files: `data.tar.gz` and <config.tar.gz>.

`data` - This directory contains the data collected and maintained by remstats. There is a sub-directory for each host and a few other special-purpose files and sub-directories:

- `ALERTS` - This file contains the alert status of all variables which have an alert associated with them, for all RRDs and hosts. It is maintained by the *alert-monitor* and will be re-created if it is missing.
- `IP_CACHE` - This is a cached mapping between IP numbers and host-names created by the routine `get_ip` via `read_ip_cache` and `write_ip_cache`. In normall use of remstats, it will be created by ping-collector and used by subsequent programs invoked during a run of *run-remstats*. It exists to reduce DNS queries and to provide consistency during a run-remstats run.
- `LAST` - This directory contains a file for each collector, named for the collector, containing the data collected on it‘s last invocation. It‘s mainly intended for debugging, but could be used for other purposes. It also contains the file `CONFIGCHANGE` which tracks when the configuration directory tree last changed.
- `LOGS` - This directory contains the event-logs, updated by various programs which call the routine `logit`, and displayed by *showlog.cgi|chowlog-cgi*.
- `NT` - This directory contains information collected by *nt-discover*, specifically: nthostinfo (which contains SRVINFO data from the *nt-status-collector*), nthosts (which contains a list of discovered NT hosts), and ntusers (which contains login information about NT users).
- `TRACEROUTES` - This contains information collected by *do-traceroutes*. Each file

(PATH-yyyymmdd-hhmmss) within it is the output of *make-path* run on the output of *traceroute* for all the hosts. There are also two other files: `PATHS` contains the latest available path for each host, and `NAME` contains the name of the latest PATH-* file.

`tmp` - Temporary files not needed over the long term. There are lock-files (`LOCK-programname`) and files used to create the lock-files (`LOCK-programname.pid`). There are logs from the current and previous runs of run-remstats (`LOG-run-remstats` and `LOG-run-remstats.old`). `STATUS-run-remstats` is updated by run-remstats to show its status while running, to be displayed by *remstats-monitor*. Nt-discover creates some files here prefixed with its name. The most important file here is `uphosts`, created by the ping-collector to contain a list of hosts which were responding to a ping on this run. This is used by the various collectors unless they are invoked without the "`-u`" flag, to allow them to skip querying hosts which aren't up.

Functions

FIXME

There are a number of remstats-supplied functions which may be useful in munging collected data for user-defined RRDs. The `data` directive in an RRD definition allows you to specify a function like:

```
data myvar=&myfunc(collector-var-name) ...
```

The functions which might be of use are:

siunits - converts large numbers to short strings. E.G. "`1200000`" would be rendered as "`1.2M`".

timestamp - converts a unix timestamp (seconds since Jan 1 1970) into a more comprehensible form "`YYYY-MM-DD hh:mm:ss`".

timestamp2 - converts a unix timestamp (seconds since Jan 1 1970) into a different, more comprehensible form "`YYYYMMDD-hhmmss`".

si_to_number - converts SI units (as produced by `siunits` above to a bare number.

m_or_km_to_km - converts a number of meters or kilometers to kilometers. Specifically, "`1.2km`" is converted to "`1.2`", but "`2500m`" is converted to "`2.5`".

sec_to_dhms - converts a number of seconds into a string like "`2d12:10:14`", meaning "2 days 12 hours 10 minutes and 14 seconds".

to_filename - the remstats file-name numger, it converts a string to one containing no characters which will give problems in file-names, by removing such characters.

cisco_modem_protocol - converts protocol names from a Cisco Access Server to numbers.

cisco_modem_modulation - converts modulation names from a Cisco Access Server to numbers.

cisco_modem_state - converts modem state names from a Cisco Access Server to numbers.

apcups_battery_status - converts battery status numbers from an APCC UPS (from SNMP) to a string indicating the status.

snmpiftype - convert an SNMP interface type to a short name.

snmpifstatus - convert an SNMP interface status to a short string.

to_ifname - munge an interface name to lower-case and remove all characters except letters, digits, colon (:), underscore (_), hyphen (-) and period (.). This is not intended to make a safe file-name (note the colon), but rather to get names that can be dealt with more easily.

If you want to add your own functions, use the *private.pl|private* file.

genindex - make an index from output of podhtml

Usage:

genindex version 1.7 from remstats 1.0.13b usage: genindex [options] file ... where options are:

```
-d          enable debugging output
-f fff  use 'fff' format for output (html, pod or text)[pod]
-h          show this help
```

Description:

genindex reads the index output of *podhtml* and builds a crude index. It's mainly for using your browser's find command on and it's not pretty. Show me something simple and better and I'll use it.

genmenu - generate a collapsing menu

Usage:

genmenu version 1.7 from remstats 1.0.13b usage: ../genmenu [options] pagename menufile where options are:

-d	enable debugging output
-h	show this help

Description:

genmenu reads the menu description file and generates a vertical menu-bar, collapsed according to which pagename you gave it. This requires all the documentation to be rebuilt whenever you change the menu definition, but avoids having to use JavaScript.
I couldn't find a simple stand-alone program that did this, so here you are. It doesn't require remstats.

The Menu Definition File

The file has a simple format. Blank lines and lines beginning with '#' are ignored. The other lines look like:

```
[tabs]pagename [page title]
```

The number of tabs shows the level of sub-menus, making the definition file easy to grasp at a glance. Note that the tabs are actual tab characters. The page title (optional) is what shows up in the menu, while the pagename is used to make the URL to link to. If the page name is xyz, then a link to xyz.html is produced. If the page title is missing, then the pagename is used instead.

podhtml - convert HTML to POD

Usage:

```
htmlpod htmlfile >podfile
```

Description:

Quick and Dirty.
I only wrote it to do the first cut for converting my old html-based documentation to pod format. It's by no means complete or even correct. However, it did convert over 90% of the HTML markup that I was using.
Use it if you want, but don't complain about it without providing a patch to fix your complaint.

lockfile

Usage:

lockfile version 1.6 from remstats 1.0.13b usage: ../lockfile [options] where options are:

-b bbb	break locks that are more than 'bbb' seconds old [-1]
-d nnn	enable debugging output at level 'nnn'
-f fff	use 'fff' for config-dir [/home/remstats/etc/config]
-F	remove the lock-file and try to lock it (force it)
-h	show this help
-r rrr	retry the lock 'rrr' times (always at least once) [0]
-s sss	sleep for 'sss' seconds between tries [10]
-u	remove the lock-file and exit

Description:

This makes remstats generic make_lockfile function available to shell scripts so that external things (like backups) can lock the same way remstats does internally, for the few things which need locking.
If the specified lockfile has no '/' in it, then it will be augmented to

```
<tempdir>/LOCK-<whatever>
```

where <tempdir> is the remstats configuration-supplied tempdir and <whatever> is what you specified as a lock-file. The remstats programs which use the make_lockfile routine, which is the heart of lockfile, all use their name as the lock-file.

podhtml - translate a POD file to HTML

Usage:

podhtml version 1.9 from remstats 1.0.13b usage: podhtml [options] podfile where options are:

-d ddd enable debugging output
-h show this help
-s sss use 'sss' as the suffix for html files [.html]
-u uuu use 'uuu' as a URL prefix

Description:

See the docs for `pod2html`. The only changes that I made intentionally from how `pod2html` does things are:

if a line looks blank it's treated as blank. I prefer to avoid surprises.

I added a new `=exec` which executes a command line and inserts the output of stdout into the resulting HTML as a <PRE> section. This was so that I could get the latest usage message from programs inserted without having to run each program separately, save its output in a file, and manually insert the file into the POD file.

I also caused it to append to a file called `podhtml–rawindex` for each =head1 and =head2, a URL for that page and section and the contents of that =headN. This is used by *genindex* to make an index.

I wrote this version in frustration with the way pod2html does links. Or doesn't. I could never tell without trying whether it would generate a link or not.

podlatex - translate a POD file to LaTeX

Usage:

podlatex version 1.6 from remstats 1.0.13b usage: podlatex [options] podfile where options are:

-d ddd enable debugging output
-h show this help
-s sss use 'sss' as the suffix for html files [.html]
-u uuu use 'uuu' as a URL prefix

Description:

See the docs for `pod2latex`. The only changes that I made intentionally from how `pod2latex` does things are:

if a line looks blank it's treated as blank. I prefer to avoid surprises.

I added a new `=exec` which executes a command line and inserts the output of stdout into the resulting HTML as a <PRE> section. This was so that I could get the latest usage message from programs inserted without having to run each program separately, save its output in a file, and manually insert the file into the POD file.

I changed the text wrapping links.

I wrote this version in frustration with the way pod2latex does links.

podpdf - translate a POD file to pdf

Usage:

```
../podpdf
[ --help --verbose <1|2 --paper <usletter> --podfile <file> ] <file>

--help
    displays this explanation of correct usage

--vebose <1|2>
    regulates the volume of progress comments: argument must be 1 or 2

--podfile <file>
    supplies the input file to process as an explicit parameter. The
    input file may also be supplied from STDIN or from the command
```

```
        line as the array element --paper.

    Further information can be found in the POD section of Pod.pm. Enter:
        perl -e "use Pod::Pdf; pod2pdf('<your_library_path>/Pod/Pdf.pm')"
    to get the POD in PDF format :)
```

Description:

See the docs for `Pod::PDF`. This is only a tiny wrapper for it.

remstats-backup

Usage:

remstats-backup

Description:

It *locks out|lockfile* run-remstats and backs up the data sub-tree (/home/remstats/data), unlocks it and then backs up the config sub-tree (/home/remstats/etc/config). It simply runs tar on the sub-trees and puts the resultant tarballs in /home/remstats/backup as data.tar.gz and config.tar.gz.

remstats-cleanup - removes stale, old files

Usage:

remstats-cleanup version 1.2 from remstats 1.0.13b usage: ../remstats-cleanup [options] where options are:

-d nnn enable debugging output at level 'nnn'
-f fff use 'fff' for config-dir [/home/remstats/etc/config]
-h show this help

Description:

It removes old collector data from /home/remstats/data/LAST, old logs from /home/remstats/data/LOGS, old traceroute data from /home/remstats/data/TRACEROUTES and old images from all the host subdirectories of /home/remstats/html.
Run it out of cron every now and then, say once a day, with a line like:

```
0 2 * * * /home/remstats/bin/remstats-cleanup
```

remstats-version

Usage:

remstats-version

Description:

It just shows the version of the installed remstats. (Which should be 1.0.13a for this installation.)

rename-host

Usage:

rename-host version 1.1 from remstats 1.0.13b usage: ../rename-host [options] where options are:

-d nnn enable debugging output at level 'nnn'
-h show this help
-i iii change the IP number in the new config-file to 'iii'

Description:

Does what's necessary to rename a monitored host, specifically:

- renames the host config-file to `oldname~`, disabling it
- renames the host's data directory to `newname`
- renames the host's html directories to `newname` (both generated CGI pages and images directories)
- copies the host config-file to `newname~`, optionally changing the IP number of the host
- renames the host configfile to `newname`

rrd-report - display summaries of an RRD file

Usage:

rrd-report version 1.12 from remstats 1.0.13a usage: rrd-report [options] rrd-file where options are:

```
-b bbb      begin at time 'bbb' (see Note 3)
-c ccc  select data from consolidation-function 'ccc' [AVERAGE]
-d ddd  enable debugging output at level 'ddd'
-D DDD  show the dates as 'DDD' [both,pretty]
                (none|both|start|finish),(raw|simple|pretty)
-e eee  end at time 'eee' (see Note 3)
-f fff  use report format 'fff' [simple]
                (from 'simple', 'label', 'html')
-h      show this help
-i iii  report on intervals 'iii' (see Note 2) [1d]
-l      list the DS names in this rrd, no report
-n nnn  use 'nnn' as the format to print the numbers [%lf]
-s sss  summary on interval 'sss' (see Note 2) [1w]
-v vvv  show (v)ariables 'vvv' (comma-separated) [ALL]
```

Note: if report interval and summary interval are equal,
no summary reporting is done.
Note 2: intervals are numbers of seconds, minutes, hours, days, weeks, Months
or years. E.G. "4w" for 4 weeks, "1M" for one month.
Note 3: Begin and end times are unix time stamp (seconds since Jan 1, 1970) or
plus-or-minus an interval, as in Note 2. E.G. "-1w" means "one week ago".

Examples:

I hope that the above is enough to use it after seeing a few examples. Here's the equivalent of the command that created the RRD for the example.

```
rrdtool create ping.rrd \
        DS:sent:GAUGE:600:0:10 \
        DS:rcvd:GAUGE:600:0:10 \
        DS:min:GAUGE:600:U:U \
        DS:avg:GAUGE:600:U:U \
        DS:max:GAUGE:600:U:U \
        RRA:AVERAGE:0.1:1:600 \
        RRA:AVERAGE:0.1:7:300 \
        RRA:AVERAGE:0.1:30:300 \
        RRA:AVERAGE:0.1:90:300 \
        RRA:AVERAGE:0.1:365:300 \
        RRA:MIN:0.1:1:600 \
        RRA:MIN:0.1:7:300 \
        RRA:MIN:0.1:30:300 \
        RRA:MIN:0.1:90:300 \
        RRA:MIN:0.1:365:300 \
        RRA:MAX:0.1:1:600 \
        RRA:MAX:0.1:7:300 \
        RRA:MAX:0.1:30:300 \
        RRA:MAX:0.1:90:300 \
        RRA:MAX:0.1:365:300
```

See "man rrdcreate" for an explanation for the command itself. The fields are:

sent/rcvd - number of ping packets sent/received

min/avg/max - the round-trip-time (min, average and max) for the pings

Here's a default report from one of my ping RRDs:

```
%rrd-report ping.rrd
[snip]
data 1999-10-25 17:20:49 1999-10-26 17:20:49 10.000000 10.000000 10.000000
data 1999-10-26 17:20:49 1999-10-27 17:20:49 10.000000 10.000000 10.000000
summary 1999-10-19 17:20:49 1999-10-26 17:20:49 10.000000 10.000000 10.000
[snip]
data 1999-11-17 16:20:49 1999-11-18 16:20:49 10.000000 10.000000 10.000000
summary 1999-11-16 16:20:49 1999-11-18 16:20:49 10.000000 10.000000 10.000
overall 1999-10-19 17:20:49 1999-11-18 16:20:49 9.978889 9.999918 10.000000
```

Each "data" line is a report for the interval covered by the two timestamps, (by default one day). The values are the requested (or in this case all) DS:CF combinations. The "summary" lines are just reports over a longer interval (by default one week). The "overall" line is for the whole selected time-period.
Hmm. There's much too much there. What I'd really like to see is just the interesting stuff. I know how many pings I'm sending during this period (10), so drop that and just show the minimum min average avg and maximum max:

```
% rrd-report -v rcvd:AVERAGE,min:MIN,avg:AVERAGE,max:MAX
data 1999-10-19 17:54:57 1999-10-20 17:54:57 9.820267 38.317778 43.948411
data 1999-10-20 17:54:57 1999-10-21 17:54:57 9.966716 39.303333 46.180111
data 1999-10-21 17:54:57 1999-10-22 17:54:57 9.907440 40.469000 48.496274
data 1999-10-22 17:54:57 1999-10-23 17:54:57 9.977827 40.232333 47.571133
[snip]
summary 1999-11-09 16:54:57 1999-11-16 16:54:57 9.950836 39.310056 52.5789
data 1999-11-17 16:54:57 1999-11-18 16:54:57 9.934164 36.400000 49.606736
summary 1999-11-16 16:54:57 1999-11-18 16:54:57 9.928672 38.285714 49.4897
overall 1999-10-19 17:54:57 1999-11-18 16:54:57 9.876767 6.194333 48.84274
```

Well, I can figure out when the period ended, so leave out the end-time, and I don't like seeing all those meaningless (in this case) decimal places, so how about:

```
% rrd-report -D start,pretty -n %.1lf -v rcvd:AVERAGE,min:MIN,avg:AVERAGE,
[snip]
data 1999-11-14 17:27:04 10.0 40.0 49.4 88.7
data 1999-11-15 17:27:04 9.7 21.7 48.0 63.4
data 1999-11-16 17:27:04 9.9 38.3 49.4 61.3
summary 1999-11-09 17:27:04 10.0 39.3 52.6 179.6
data 1999-11-17 17:27:04 9.9 36.4 49.6 117.2
summary 1999-11-16 17:27:04 9.9 38.3 49.5 65.9
overall 1999-10-19 18:27:04 9.9 6.2 48.8 179.6
```

OK. I'd like to see the last year with a one-week interval, with no summaries. (Setting the report-interval to the same as the summary-interval drops summaries. You still get an overall line.)

```
% rrd-report -D start,pretty -n %.1lf -v rcvd:AVERAGE,min:MIN,avg:AVERAGE,
x:MAX -i 1w -s 1w
data 1998-11-19 09:04:43 NODATA NODATA NODATA NODATA
[snip]
data 1999-02-25 09:04:43 9.9 45.0 55.0 64.2
data 1999-03-04 09:04:43 10.0 43.9 54.5 64.3
[snip]
data 1999-11-11 09:04:43 10.0 39.3 51.7 179.6
data 1999-11-18 09:04:43 9.9 37.4 49.7 169.7
overall 1998-11-19 09:04:43 9.6 0.0 45.3 103.7
```

And for those of us who like to see it on the web:

```
<table border=1> <tr>
<th>rcvd:AVERAGE</th> <th>min:MIN</th> <th>avg:AVERAG <th>max:MAX</th> </tr> <tr>
        <td>1998-11-19 09:06:37</td>
        <td align=right> </td>
        <td align=right> </td>
        <td align=right> </td>
</tr> <tr>  <td align=right> </td>
        <td>1998-11-26 09:06:37</td>
        <td align=right> </td>
        <td align=right> </td>
        <td align=right> </td>
</tr> <tr>  <td align=right> </td>
        <td>1998-12-03 09:06:37</td>
        <td align=right> </td>
        <td align=right> </td>
        <td align=right> </td>
```

```
</tr> <tr>  <td align=right> </td>
            <td>1998-12-10 09:06:37</td>
            <td align=right> </td>
            <td align=right> </td>
            <td align=right> </td>
</tr> <tr>  <td align=right> </td>
            <td>1998-12-17 09:06:37</td>
            <td align=right> </td>
            <td align=right> </td>
            <td align=right> </td>
</tr> <tr>  <td align=right> </td>
            <td>1998-12-24 09:06:37</td>
            <td align=right> </td>
            <td align=right> </td>
            <td align=right> </td>
</tr> <tr>  <td align=right> </td>
            <td>1998-12-31 09:06:37</td>
            <td align=right> </td>
            <td align=right> </td>
            <td align=right> </td>
</tr> <tr>  <td align=right> </td>
            <td>1999-01-07 09:06:37</td>
            <td align=right> </td>
            <td align=right> </td>
            <td align=right> </td>
</tr> <tr>  <td align=right> </td>
            <td>1999-01-14 09:06:37</td>
            <td align=right> </td>
            <td align=right> </td>
            <td align=right> </td>
</tr> <tr>  <td align=right> </td>
            <td>1999-01-21 09:06:37</td>
            <td align=right> </td>
            <td align=right> </td>
            <td align=right> </td>
</tr> <tr>  <td align=right> </td>
            <td>1999-01-28 09:06:37</td>
            <td align=right> </td>
            <td align=right> </td>
            <td align=right> </td>
</tr> <tr>  <td align=right> </td>
            <td>1999-02-04 09:06:37</td>
            <td align=right> </td>
            <td align=right> </td>
            <td align=right> </td>
</tr> <tr>  <td align=right> </td>
            <td>1999-02-11 09:06:37</td>
            <td align=right> </td>
            <td align=right> </td>
            <td align=right> </td>
</tr> <tr>  <td align=right> </td>
            <td>1999-02-18 09:06:37</td>
            <td align=right> </td>
            <td align=right> </td>
            <td align=right> </td>
</tr> <tr>  <td align=right> </td>
            <td>1999-02-25 09:06:37</td>
            <td align=right>9.9</td>
            <td align=right>45.0</td>
```

```
                <td align=right>55.0</td>
</tr> <tr>      <td align=right>64.2</td>
                <td>1999-03-04 09:06:37</td>
                <td align=right>10.0</td>
                <td align=right>43.9</td>
                <td align=right>54.5</td>
</tr> <tr>      <td align=right>64.3</td>
                <td>1999-03-11 09:06:37</td>
                <td align=right>10.0</td>
                <td align=right>41.6</td>
                <td align=right>54.1</td>
</tr> <tr>      <td align=right>103.7</td>
                <td>1999-03-18 09:06:37</td>
                <td align=right>9.4</td>
                <td align=right>30.7</td>
                <td align=right>49.8</td>
</tr> <tr>      <td align=right>62.0</td>
                <td>1999-03-25 09:06:37</td>
                <td align=right>6.1</td>
                <td align=right>0.0</td>
                <td align=right>31.9</td>
</tr> <tr>      <td align=right>60.5</td>
                <td>1999-04-01 09:06:37</td>
                <td align=right>1.7</td>
                <td align=right>0.0</td>
                <td align=right>8.5</td>
</tr> <tr>      <td align=right>52.4</td>
                <td>1999-04-08 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>47.2</td>
                <td align=right>49.8</td>
</tr> <tr>      <td align=right>52.4</td>
                <td>1999-04-15 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>48.2</td>
                <td align=right>49.7</td>
</tr> <tr>      <td align=right>52.4</td>
                <td>1999-04-22 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>48.0</td>
                <td align=right>49.7</td>
</tr> <tr>      <td align=right>52.1</td>
                <td>1999-04-29 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>47.6</td>
                <td align=right>49.8</td>
</tr> <tr>      <td align=right>52.7</td>
                <td>1999-05-06 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>46.7</td>
                <td align=right>49.7</td>
</tr> <tr>      <td align=right>60.3</td>
                <td>1999-05-13 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>46.7</td>
                <td align=right>51.9</td>
</tr> <tr>      <td align=right>90.4</td>
                <td>1999-05-20 10:06:37</td>
                <td align=right>10.0</td>
```

```
                <td align=right>48.1</td>
                <td align=right>53.5</td>
</tr> <tr>      <td align=right>100.4</td>
                <td>1999-05-27 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>40.7</td>
                <td align=right>50.0</td>
</tr> <tr>      <td align=right>93.7</td>
                <td>1999-06-03 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>40.0</td>
                <td align=right>44.3</td>
</tr> <tr>      <td align=right>54.7</td>
                <td>1999-06-10 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>39.9</td>
                <td align=right>41.2</td>
</tr> <tr>      <td align=right>56.3</td>
                <td>1999-06-17 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>40.0</td>
                <td align=right>41.1</td>
</tr> <tr>      <td align=right>56.3</td>
                <td>1999-06-24 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>39.0</td>
                <td align=right>41.1</td>
</tr> <tr>      <td align=right>50.7</td>
                <td>1999-07-01 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>39.5</td>
                <td align=right>40.5</td>
</tr> <tr>      <td align=right>48.2</td>
                <td>1999-07-08 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>40.0</td>
                <td align=right>40.7</td>
</tr> <tr>      <td align=right>44.6</td>
                <td>1999-07-15 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>41.0</td>
                <td align=right>42.6</td>
</tr> <tr>      <td align=right>48.2</td>
                <td>1999-07-22 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>43.0</td>
                <td align=right>43.3</td>
</tr> <tr>      <td align=right>48.2</td>
                <td>1999-07-29 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>41.0</td>
                <td align=right>42.5</td>
</tr> <tr>      <td align=right>61.6</td>
                <td>1999-08-05 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>40.9</td>
                <td align=right>41.8</td>
</tr> <tr>      <td align=right>61.6</td>
                <td>1999-08-12 10:06:37</td>
```

```
                <td align=right>9.6</td>
                <td align=right>34.5</td>
                <td align=right>39.7</td>
</tr> <tr>      <td align=right>47.0</td>
                <td>1999-08-19 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>39.9</td>
                <td align=right>40.3</td>
</tr> <tr>      <td align=right>46.3</td>
                <td>1999-08-26 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>39.9</td>
                <td align=right>40.3</td>
</tr> <tr>      <td align=right>48.0</td>
                <td>1999-09-02 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>39.6</td>
                <td align=right>40.6</td>
</tr> <tr>      <td align=right>54.9</td>
                <td>1999-09-09 10:06:37</td>
                <td align=right>9.8</td>
                <td align=right>39.9</td>
                <td align=right>48.0</td>
</tr> <tr>      <td align=right>70.5</td>
                <td>1999-09-16 10:06:37</td>
                <td align=right>9.9</td>
                <td align=right>44.5</td>
                <td align=right>50.0</td>
</tr> <tr>      <td align=right>58.1</td>
                <td>1999-09-23 10:06:37</td>
                <td align=right>9.9</td>
                <td align=right>46.4</td>
                <td align=right>49.9</td>
</tr> <tr>      <td align=right>58.0</td>
                <td>1999-09-30 10:06:37</td>
                <td align=right>9.9</td>
                <td align=right>46.0</td>
                <td align=right>50.1</td>
</tr> <tr>      <td align=right>62.3</td>
                <td>1999-10-07 10:06:37</td>
                <td align=right>9.9</td>
                <td align=right>46.8</td>
                <td align=right>50.0</td>
</tr> <tr>      <td align=right>56.9</td>
                <td>1999-10-14 10:06:37</td>
                <td align=right>9.8</td>
                <td align=right>39.1</td>
                <td align=right>47.1</td>
</tr> <tr>      <td align=right>56.4</td>
                <td>1999-10-21 10:06:37</td>
                <td align=right>10.0</td>
                <td align=right>38.9</td>
                <td align=right>47.0</td>
</tr> <tr>      <td align=right>59.9</td>
                <td>1999-10-28 10:06:37</td>
                <td align=right>9.7</td>
                <td align=right>6.2</td>
                <td align=right>47.0</td>
                <td align=right>59.1</td>
```

```
</tr> <tr>
                <td>1999-11-04 09:06:37</td>
                <td align=right>9.9</td>
                <td align=right>46.0</td>
                <td align=right>50.8</td>
</tr> <tr>      <td align=right>126.1</td>
                <td>1999-11-11 09:06:37</td>
                <td align=right>10.0</td>
                <td align=right>39.3</td>
                <td align=right>51.7</td>
</tr> <tr>      <td align=right>179.6</td>
                <td>1999-11-18 09:06:37</td>
                <td align=right>9.9</td>
                <td align=right>37.4</td>
                <td align=right>49.7</td>
</tr> <tr>      <td align=right>169.7</td>
                <td>1998-11-19 09:06:37</td>
                <td align=right>9.6</td>
                <td align=right>0.0</td>
                <td align=right>45.3</td>
</tr> </table><td align=right>103.7</td>
```

[You'll just have to look in the web version of this documentation to see what it looks like.]

[You'll just have to look in the web version of this documentation to see what it looks like.]

rt-updater

Usage:

rt-updater version 1.1 from remstats 1.0.13b usage: ../rt-updater [options] host rrd graph temp-rrdfile rrdcgi-file where options are:

```
 -a aaa  keep 'aaa' samples in the archive [720]
 -A      auto-set step from max collector test run-time
 -c ccc  use consolidation functions 'ccc' (comma-separated list) in
         creating the RRAs [AVERAGE,MIN,MAX]
 -d nnn  enable debugging output at level 'nnn'
 -e      use existing rrd file
 -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
 -F      run forever; usually runs until the archive is full
 -h      show this help
 -r rrr  make the rrdcgi page refresh every 'rrr' seconds [step]
 -s sss  update step time of 'sss' seconds [5]
 -t ttt  how many times to test the collector [3]
 -u      unlink rrd-file, rrdcgi-file and image files after run
```

Note: 'temp-rrdfile' must be a full path to the file or the generated 'rrdcgi-file' will not work.

Description:

[This is not well integrated with the rest of remstats. If I figure out a good way to do it I will, but so far ...]

This program creates an RRD file, using the specified remstats RRD definition, but with a different set of RRAs and step time. It creates a command pipeline to run the required (RRD-specified) *collector|collectors* into the *updater*. It then runs this a number of times, to make sure that it can obtain data fast enough to meet the user-specified step time. It writes an `rrdcgi` web-page to show the user-specified graph from that RRD. Finally, it loops, running the collector-to-updater pipeline. This process will run until it fills the archive, or forever, if requested.

The actual image files are stored in `/home/remstats/html/GRAPHS/TMP`, which should be created as writable by both the `@@USER@@` user and the `apache` group, and set-group-id to the `apache` group. (This should be done automatically as part of the install and running the *page-writer*.) This will allow the rrdcgi web-page to write the image files there while running as the web-server's user.

This is intended for monitoring a few things during testing of something, to create very fine-grained data not to replace remstats usual processes, as it's much more CPU intensive.

Thank-you‘s

```
Tobias Oetiker
Larry Wall and the rest of the perl hackers
Vikas Aggarwal
Ehud Gavron
Will Maton
Adam Kennedy
Ken Filipps
Andrew Cochran
Marek Snowarski
Matt Duggan
Steve Francis
Alexander Reelsen
Jon Villarreal
Robert Jordens
Vincent Forgues
Anik Richard
Arnauld Michelizza
```

I‘ve probably forgotten some others. Please remind me if I‘ve forgotten your contribution.