

Paragon Active Assurance Operations Guide

Published
2022-05-19

RELEASE
3.3

Table of Contents

Introduction
Finding Out Your Paragon Active Assurance Software Version
Backing Up Product Data
Restoring Product Data from Backup
Updating Databases with New User Passwords
PostgreSQL Configuration
Managing Control Center and NETCONF Users
Importing and Exporting Test and Monitor Templates
Monitoring the System
Tuning the System
Troubleshooting
Deleting Licenses from the System
Deleting Paragon Active Assurance Accounts
Uninstalling Paragon Active Assurance

Introduction

The Operations Guide covers various aspects of operating and maintaining an on-premise installation of Paragon Active Assurance. Among the topics in this guide are the following:

- Data backup and restore
- Updating user passwords
- Managing authentication
- Monitoring the system
- Tuning the system

Please note that troubleshooting is covered in the *Installation Guide*.

Finding Out Your Paragon Active Assurance Software Version

To find out what version of Paragon Active Assurance you currently have installed, you can use this command:

```
dpkg -l | grep paa
```

Backing Up Product Data

IN THIS SECTION

- [Automated Backup: Using the `ncc backup` Command | 2](#)
- [Troubleshooting: Manual Backup | 3](#)

Automated Backup: Using the `ncc backup` Command

To make a complete backup of your Paragon Active Assurance system, use the command

```
ncc backup
```

This command takes backups of all the following:

- Main PostgreSQL database
- Plugin service PostgreSQL database
- Plugin service files
- RRD files
- TimescaleDB database
- TimescaleDB configuration files
- OpenVPN keys
- Control Center configuration files
- Licenses

Options:

- `--without-rrd` creates a backup without the RRD files (`paa_cc_rrd.tar.gz`)
- `--without-timescaledb` creates a backup without the TimescaleDB database (`paa_cc_timescaledb.tar.gz`)
- `--without-configs` creates a backup without the Control Center configuration files (`paa_cc_configs.tar.gz`)
- `--without-db` creates a backup without the main and plugin service databases (`paa_cc_postgres.sql` and `paa_cc_plugins.sql`)

The output file is a tarball named according to the pattern `paa_backup-version-yyyy-mm-dd_hh-mm-ss.tar.gz`.

These files serve as input to restoring the system with the `ncc restore` command; see the chapter ["Restoring Product Data from Backup" on page 6](#).

Troubleshooting: Manual Backup

Should the `ncc backup` command fail for some reason, you can achieve the same result manually by following the steps in this section.

To make a consistent backup of your Paragon Active Assurance data, you need to stop all services accessing the database before starting the backup procedure. It is possible to make backups of a live system, but data consistency cannot then be guaranteed.

Proceed as follows:

- **Stop** Paragon Active Assurance services:

```
sudo ncc services stop
```

- **Make backups** according to the subsections below.
- **Start** Paragon Active Assurance services:

```
sudo ncc services start
```

Backing Up the Main PostgreSQL Database and Plugin Database

The plugin database is created along with the main PostgreSQL database, but it needs to be backed up separately.

Run these commands:

```
pg_dump -h localhost -U netrounds netrounds > cc_postgres.sql  
pg_dump -h localhost -U netrounds paa-plugins > paa_plugins.sql
```

NOTE: The `pg_dump` command will ask for a password which can be found in `/etc/netrounds/netrounds.conf` under "postgres database". The default password is "netrounds".

To learn about advanced options of `pg_dump`, such as compression, type

```
pg_dump --help
```

NOTE: Alternatively, you may want to back up the database in binary format. If so, use this command:

```
pg_dump -h localhost -U netrounds -Fc netrounds > cc_postgres.binary
pg_dump -h localhost -U netrounds -Fc paa-plugins > paa_plugins.binary
```

Backing Up the RRD Files

The RRD (round-robin database) files contain the Paragon Active Assurance measurement data.

- For a small-scale setup (< 50 GB), use this backup command:

```
sudo tar -czf cc_rrd.tar.gz /var/lib/netrounds/rrd
```

- For a large-scale setup (> 50 GB), making a tarball of the RRD files might take too long, and taking a snapshot of the volume can be a better idea. Possible solutions for doing this include: using a file system that supports snapshots, or taking a snapshot of the virtual volume if the server is running in a virtual environment.

Backing Up Plugin Files

Back up plugin files with the commands

```
sudo tar -czf paa_plugins.tar.gz /var/lib/netrounds/plugins
sudo tar -czf paa_plugin_keys.tar.gz /etc/netrounds/plugin_keys
```

Backing Up the TimescaleDB Database

To back up the TimescaleDB database, you must first start it.

Backup of the TimescaleDB database also requires the `timescaledb` service related to the Streaming API to be enabled and running:

```
sudo ncc services enable timescaledb
sudo ncc services start timescaledb
```

For more information on this API, see the document *Streaming API Guide*.

To back up the TimescaleDB database manually, run:

```
sudo /etc/netrounds/backup-restore-timescaledb.sh --backup <backup dir>
```

Backups of TimescaleDB data are stored in `/var/lib/netrounds/rrd/timescaledb/pgbackrest/repo/<backup dir>`.

Backing Up the OpenVPN Keys

Use this command:

```
sudo tar -czf cc_openvpn.tar.gz /var/lib/netrounds/openvpn
```

Backing Up the Licenses

Use this command:

```
sudo tar -czf cc_license.tar.gz /var/lib/netrounds/license
```

Backing Up the Configuration Files

Make copies of the following files:

- `/etc/apache2/sites-available/netrounds-ssl.conf`
- `/etc/apache2/sites-available/netrounds.conf`
- `/etc/netrounds/consolidated.conf`
- `/etc/netrounds/metrics.yaml`
- `/etc/netrounds/netrounds.conf`
- `/etc/netrounds/plugin.yaml`
- `/etc/netrounds/probe-connect.conf`
- `/etc/netrounds/restol.conf`
- `/etc/netrounds/test-agent-gateway.yaml`
- `/etc/netrounds/timescaledb.conf`

- `/etc/openvpn/netrounds.conf`

For example:

```
cp /etc/apache2/sites-available/netrounds-ssl.conf /etc/apache2/sites-available/netrounds-ssl.conf.old
```

Restoring Product Data from Backup

IN THIS SECTION

- [Automated Restore: Using the `ncc restore` Command | 6](#)
- [Troubleshooting: Manual Restore | 8](#)

Automated Restore: Using the `ncc restore` Command

To restore a Paragon Active Assurance system from a backup taken with the `ncc backup` command, run this command on the backup tarball:

```
ncc restore paa_backup-yyyy-mm-dd_hh-mm-ss.tar.gz
```

This command does all of the following:

- Drops, recreates, and restores the main PostgreSQL database
- Drops, recreates, and restores the plugin PostgreSQL database
- Restores the plugin tarballs and signing keys
- Removes any existing RRDs, and restores RRD files
- Restores the TimescaleDB database and TimescaleDB data
- Restores TimescaleDB configuration files
- Removes any existing OpenVPN keys, and restores OpenVPN keys

- Restores Control Center configuration files
- Activates licenses

Option:

- `--timescaledb-in-place`: By default the `ncc restore` command assumes that you are doing the restore on a different host. If you want to perform an in-place restore of the TimescaleDB database, add this option to the command.

Activating a New License for the Restored System

If you are restoring the system on a new host, or on the same host but with a different UUID, the system requires a new license. Please see the Installation Guide, chapter Getting Started with Paragon Active Assurance, for details on how to obtain and activate a license.

Performing a Partial Restore

You can optionally leave out certain parts of the backup from the restore operation.

Only backup files that are not mandatory can be left out of the restore.

The following files can be omitted freely:

- `paa-cc-backup.log`
- `paa_cc_rrd.tar.gz`
- `paa_cc_configs.tar.gz`

The following files can both be left out, but one is not valid without the other:

- `paa_cc_postgres.sql`
- `paa_cc_plugins.sql`

Proceed in the following steps (the backup version and timestamp are suppressed below for readability):

- List the backup tarball content in order to identify parts to exclude:

```
tar -tf paa-cc-backup-<...>.tar.gz
```

- Unpack the backup tarball, excluding files as desired. Below is an example where the RRD files are excluded:

```
tar -zxvf paa-cc-backup-<...>.tar.gz --exclude "paa-cc-backup-<...>/paa_cc_rrd.tar.gz"
```

- Rename the backup folder to match the intended final tarball name:

```
mv paa-cc-backup-<...> paa-cc-backup-without-rrd-<...>
```

- Create a new tarball from the remaining files:

```
tar -pczf paa-cc-backup-without-rrd-<...>.tar.gz paa-cc-backup-without-rrd-<...>/
```

Troubleshooting: Manual Restore

Should the `ncc restore` command fail for some reason, you can restore the system manually as described in this section.

In what follows, the backup files are assumed to be named in the same way as in the chapter ["Backing Up Product Data" on page 1](#).

NOTE: The procedure that follows requires that Control Center has been installed (so that the `netrounds` database exists) and that the backup files are from the same version of Control Center as the installed version.

1. **Stop** all Paragon Active Assurance **services** that are accessing the database:

```
sudo ncc services stop
```

2. **Drop the main PostgreSQL database and plugin database:**

```
sudo -u postgres psql -c "DROP DATABASE netrounds;"
sudo -u postgres psql -c 'DROP DATABASE "paa-plugins";'
```

3. Recreate the main PostgreSQL database and plugin database:

```
sudo -u postgres psql -c "CREATE DATABASE netrounds OWNER netrounds ENCODING 'UTF8'
TEMPLATE 'template0';"
sudo -u postgres psql -c 'CREATE DATABASE "paa-plugins" OWNER netrounds ENCODING "UTF8"
TEMPLATE "template0";'
```

4. Restore the main PostgreSQL database and plugin database:

```
sudo -u postgres psql --set ON_ERROR_STOP=on netrounds < cc_postgres.sql
sudo -u postgres psql --set ON_ERROR_STOP=on paa-plugins < paa_plugins.sql
```

Setting `ON_ERROR_STOP` to on will cause psql to exit if an error occurs. For further details, see the PostgreSQL documentation: <https://www.postgresql.org/docs/9.4/backup-dump.html#BACKUP-DUMP-RESTORE>.

There are several options for the backup format. Please refer to the psql documentation (<https://www.postgresql.org/docs/>) if you are using anything other than plain .sql.

If you backed up the databases (manually) in binary format, use these commands to restore them (you will be prompted for a password which is by default "netrounds"):

```
pg_restore -Fc -h localhost -d netrounds -c --verbose --disable-triggers cc_postgres.binary
-U netrounds --password
pg_restore -Fc -h localhost -d paa-plugins -c --verbose --disable-triggers
paa_plugins.binary -U netrounds --password
```

5. Restore the plugin tarballs and signing keys:

```
sudo tar -xzf paa_plugins.tar.gz -C /
sudo tar -xzf paa_plugin_keys.tar.gz -C /
```

6. Restore the OpenVPN keys:

```
# Remove any existing OpenVPN keys
sudo rm -rf /var/lib/netrounds/openvpn
```

```
# Unpack the backed-up keys
sudo tar -xzf cc_openvpn.tar.gz -C /
```

7. **Restore RRD files**, either from a tarball or from a snapshot. Compare the chapter ["Backing Up Product Data" on page 1](#). In the tarball case, the procedure is as follows:

```
# Remove any existing RRDs
sudo rm -rf /var/lib/netrounds/rrd

# Unpack the backed-up RRDs
sudo tar -xzf cc_rrd.tar.gz -C /
```

8. Restore the TimescaleDB database:

The backup directory needs to be stored under `/var/lib/netrounds/rrd/timescaledb/pgbackrest/repo/`.

Before doing the restore, copy the `.conf` files in the data directory `/var/lib/netrounds/rrd/timescaledb/data/` to a different location. This allows you to restore any configuration changes performed after the last full backup.

- To perform an *in-place* database restore, proceed as follows:
 - Create a directory `/tmp/timescaledb_conf`.
 - Copy the `*.conf` files in `/var/lib/netrounds/rrd/timescaledb/data` to `/tmp/timescaledb_conf`.
 - Run the command

```
sudo /etc/netrounds/backup-restore-timescaledb.sh --restore <backup dir>
```

- Copy the `*.conf` files from `/tmp/timescaledb_conf` to `/var/lib/netrounds/rrd/timescaledb/data`.
- To restore a database from host A on a *different host* B, proceed as follows:
 - On host B, install and configure Control Center according to the *Installation Guide*, and enable the metrics service.
 - From host A, copy the contents of `/var/lib/netrounds/rrd/timescaledb/pgbackrest` to the same directory on host B.
 - On host B, run the command

```
sudo /etc/netrounds/backup-restore-timescaledb.sh --restore <backup dir>
```

- From host A, copy the *.conf files from /var/lib/netrounds/rrd/timescaledb/data to the same directory on host B.
9. **Copy the license files** into their original location. (Alternatively, you may download the license files once more from the Juniper EMS Portal and apply them using the `ncc license activate` command, as explained in the [Installation Guide](#).)
 10. **Start** Paragon Active Assurance services:

```
sudo ncc services start
```

Updating Databases with New User Passwords

IN THIS SECTION

- [Main PostgreSQL Database | 11](#)
- [TimescaleDB Database | 13](#)
- [Streaming API | 13](#)

Main PostgreSQL Database

The chapter "[PostgreSQL Configuration](#)" on page 13 tells how to change the password for the "netrounds" user in a configuration file.

You also need to update the password in the following files:

- /etc/netrounds/netrounds.conf
- /etc/netrounds/plugin.yaml
- /etc/netrounds/probe-connect.conf

Furthermore, you must update that user's role in the database. How to do that is explained below.

- Log in to the database using:

```
sudo -u postgres psql
```

- Inside the database, execute this command (substituting the correct new password):

```
ALTER ROLE netrounds WITH PASSWORD 'securePassword';
```

The relevant PostgreSQL documentation is found here: www.postgresql.org/docs/9.0/sql-alterrole.html

Another option is to run the following after logging in to the database:

```
\password netrounds
```

and then provide the password twice.

After changing the password in the database as well as in the configuration files, reboot the system:

```
sudo reboot
```

Using Special Characters in the Password

In the ALTER ROLE query, if the character used to surround the password string occurs in the password itself, that character must be escaped. The escape character is a single quote (').

Suppose you want to set the password #secure. To this end, run the query

```
ALTER ROLE netrounds WITH PASSWORD ''#secure';
```

However, if you run

```
\password netrounds
```

you can enter any password characters without escaping them.

TimescaleDB Database

To change the password for the paa user, do the following:

- Connect to the database as the paa user (password paa). Full details on how to do this are found in the document [Querying Metrics in TimescaleDB](#), chapter [Preparations](#).

Then run:

```
ALTER USER paa PASSWORD 'new_password';
```

- In the file `/etc/netrounds/metrics.yaml`, enter the new password in the field `db-password`. Then restart the `netrounds-metrics` service:

```
sudo systemctl start netrounds-metrics
```

To change the password for the paaread user, do the following:

- Connect to the database as the paaread user (password paaread) and run

```
ALTER USER paaread PASSWORD 'new_password';
```

Streaming API

Regarding authentication for accessing Kafka via the Streaming API, see the document [Streaming API Guide](#), chapter [Configuring the Streaming API](#).

PostgreSQL Configuration

PostgreSQL settings are found in the following configuration file:

- `/etc/netrounds/probe-connect.conf`

Below is an example of file contents specifying username "user" and password "password".

```
DATABASES['default']['USER'] = 'user'  
DATABASES['default']['PASSWORD'] = 'password'
```

When updating the password in `probe-connect.conf`, you need to escape any characters that are identical to the character surrounding strings (above, this is the single quote, `'`). It is also best to escape any other non-alphanumeric characters. The escape character used is the backslash.

You can verify that you have set the intended password as follows:

```
source /etc/netrounds/probe-connect.conf  
echo $PASSWORD
```

NOTE: After making changes to these settings, you must also update the PostgreSQL database itself with the new password. How to do that is explained in the chapter ["Updating the Database with a New User Password" on page 11](#).

Managing Control Center and NETCONF Users

IN THIS SECTION

- [Managing Control Center Users | 14](#)
- [Managing NETCONF Users | 16](#)

Managing Control Center Users

- You can **list** all currently existing users with

```
ncc user-list
```


For each user is indicated:

- the user's permission for the account (read/write/admin)
- whether the user is owner
- You can **update a user's password** with

```
ncc user-update <user email> --password <new password>
```

For available options please use --help.

- You can **activate a user** that was previously deleted or disabled with

```
ncc user-activate <user email>
```

- There is also the following command:

```
ncc user-delete <user email>
```

By default, this command **deactivates the user** so that it can no longer log in. However, the user's permissions remain, as does all data created by the user. The user can be reactivated with the `ncc user-activate` command.

The user can be completely **removed** by adding the `--force` flag. This removes the user from the database *along with all related database objects, including measurement results*. You are prompted to confirm this action.

A user that owns an account cannot be removed. The account owner must first be changed using the `account-update` command (see below).

- To **create a superuser**, use this command:

```
ncc user-create <email> --password --is-superuser
```

A superuser has access to all accounts in the system and is the only type of user authorized to perform certain tasks.

- To **change the owner of an account**, run this command:

```
ncc account-update <account short name> --owner <user email>
```

Managing NETCONF Users

These commands (as given here) are all executed from the directory `/opt/netrounds-confd/`.

- To create a user "user1" with password "pwd1" in ConfD, run:

```
./ncc-netconf user create --username=user1 --password=pwd1
```

- To list all available users in ConfD:

```
./ncc-netconf user list
```

- To update the password to "pwd2" for user "user1" in ConfD:

```
./ncc-netconf user update --username=user1 --password=pwd2
```

- To delete the user "user1" from ConfD:

```
./ncc-netconf user delete --username=user1
```

Importing and Exporting Test and Monitor Templates

IN THIS SECTION

- [Listing Templates | 17](#)
- [Exporting Templates | 17](#)
- [Importing Templates | 18](#)
- [Usage Examples | 18](#)

Templates for tests and monitors can be exported from one installation of Control Center and imported into another (or into a different account on the same Control Center). The following commands are used for this purpose:

- `ncc template` for tests
- `ncc monitor-template` for monitors

The syntax is identical for these two commands and is detailed below for `ncc template`.

Listing Templates

To list all templates with account name, template name and ID, give this command:

```
ncc template list [--account NAME]
```

Use the `--account` flag to list templates from a specific account only.

Exporting Templates

Use this command to export template configurations in JSON format:

```
ncc template export [--account NAME] [--file NAME] [id ...]
```

Here, `[id ...]` is a list of IDs of the templates you want to export. If no template IDs are specified, all templates are exported.

The `--file` flag specifies the name of the output file. If you omit this, the output will be written to stdout.

Use the `--account` flag to filter templates by account.

Importing Templates

On exporting templates, you can import them to a specified account as follows:

```
ncc template import --account NAME [--force_overwrite] [export file name]
```

Here, `export file name` is an output file obtained from `ncc template export`. (If no file is specified, the import command reads from `stdin`.)

If the `force_overwrite` flag is set, then any templates with the same name that already exist in the account will be overwritten. Using this option is **not recommended** in general. It is preferable to rename the existing template(s) before importing.

Usage Examples

- Export a single template named "template1":

```
ncc template export template1 --file my_template.json
```

- Export all templates from account "demo":

```
ncc template export --account demo --file all_demo_templates.json
```

- Export all templates on the server:

```
ncc template export --file all_templates.json
```

- Import templates from export file to account "demo":

```
ncc template import --account demo all_demo_templates.json
```

Monitoring the System

IN THIS SECTION

- [System Parameters | 19](#)
- [Application Parameters | 20](#)
- [Licenses | 20](#)
- [Services | 20](#)
- [Logs | 22](#)

This section points to parameters that are useful for monitoring the health of the Paragon Active Assurance system.

System Parameters

Standard system parameters:

- CPU utilization
 - Should stay below 80%
- Memory utilization, excluding cache and buffers
 - Should stay below 80%
- Disk utilization
 - Should stay below 80%
- Network interface load
 - Should stay below 80%

Application Parameters

For Control Center, run the command

```
ncc status
```

The following parameters are of particular interest:

- **test_agent_appliance_online:** This indicates how many Test Agent Appliances are currently logged in to the server. The desirable numbers here are of course dependent on how you deploy the Test Agents.
- **scheduled_call_latency:** This indicates how far behind schedule the background job processing is. Among other things, this processing collects results from Test Agents and generates periodic reports. The latency should stay below 10 s.

Licenses

The expiry date for the license activated on the server can be inspected with the command

```
ncc license show
```

In the output from this command, look for `end_date`.

Services

The following Paragon Active Assurance services should normally be running:

- **netrounds-callexecuter:** Handles background tasks such as
 - Periodically polling monitors: collecting measurement results, updating Test Agent configurations, and evaluating alarms
 - Running scheduled tests
 - Sending periodic reports
- **netrounds-confd:** *Runs if ConfD has been installed.* ConfD is used as an intermediary between the Paragon Active Assurance system and the NETCONF & YANG API. An orchestrator can use this API to configure network equipment and trigger measurements.

- **netrounds-consolidated:** Used to proxy internal communication between services.
- **netrounds-license-daemon:** From here Control Center learns what licenses to apply to the system. The license daemon uses the Juniper License SDK which is used across all Juniper products.
- **netrounds-metrics:** *Runs if Streaming API or TimescaleDB is enabled.* Receives metrics from Kafka, ingests these metrics into TimescaleDB, and pushes them to the Streaming API.
- **netrounds-plugin:** Plugins handles measurement tasks. This service enables plugins to be uploaded to Control Center independently of Control Center upgrades. See the in-app help under "Plugins".
- **netrounds-probe-login:** Server which exposes a set of functions to Test Agent Appliances so that they can notify this server about state changes.
- **netrounds-ta3-compat:** Needed to achieve full compatibility between Test Agent Applications and Control Center.
- **netrounds-test-agent-gateway:** Used to communicate with Test Agent Applications and forward messages between these and Kafka.
- **netrounds-timescaledb:** *Runs if Streaming API or TimescaleDB is enabled.* Time-series database extension for PostgreSQL. Uses a dedicated PostgreSQL database which is distinct from the one used for configuration.
- **openvpn@netrounds:** Used to communicate with Test Agent Appliances.

In addition, the following services related to Paragon Active Assurance should be running:

- **apache2:** Web server hosting the webapp and the REST API.
- **kafka:** Message bus for sharing data asynchronously between services.
- **ntpd:** Maintains the system time in synchronization with time servers using the Network Time Protocol (NTP).
- **postgresql:** Database for storing configuration and state for the system: basically everything except measurement results and OpenVPN keys.
- **zookeeper:** Centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. Zookeeper is relied on by Kafka.

Finally, it is a good idea to monitor the following system services which are relied on by Paragon Active Assurance:

- **cron:** Job scheduler.
- **irqbalance:** Distributes hardware interrupts across processors on a multiprocessor system.

Logs

Webapp Logs

The file `/var/log/apache/netrounds_access.log` contains all HTTP requests made to the Control Center web GUI.

The file `/var/log/apache/netrounds_error.log` contains all errors reported by Apache for HTTP requests towards the Control Center web GUI. Console output from the Control Center back-end is also in this file; by default, all logging is done by the console.

The configuration of the logging is done using the Python logging module configuration schema (docs.python.org/2/library/logging.config.html#configuration-dictionary-schema).

Call Executer Logs

To see the logs for this service, you can use `journalctl`, for example to look at the logs from the last hour and then follow all incoming logs:

```
sudo journalctl -u netrounds-callexecuter.service --since "1 hour ago" --follow
```

Plugin Service Logs

For the plugin system you can look at the logs using:

```
sudo journalctl -u netrounds-plugin.service
```

Metrics Service Logs

NOTE: These logs are only relevant if you have this feature enabled, which is the case if you are using the Streaming API or TimescaleDB.

```
sudo journalctl -u netrounds-metrics.service
```


TimescaleDB

To view TimescaleDB logs, run:

```
sudo journalctl -u netrounds-timescaledb.service
```

Kafka

To view Kafka logs, run:

```
sudo journalctl -u kafka
```

Zookeeper

To view Zookeeper logs, run:

```
sudo journalctl -u zookeeper
```

Custom Formatters

Paragon Active Assurance provides two custom formatters that yield additional information about log entries and thus should be favored over Python's default formatters:

- The formatter `netrounds.utils.loggers.ContextFormatter` provides `record_tags` and `context_tags` that provide additional information about the context of the log entry, for example what Test Agent the log entry is about. Colors are also supported.
- The formatter `netrounds.thirdparty.logstash_formatter.ContextLogstashFormatterV1` outputs a JSON format compatible with Logstash.

The custom formatters are added by default in:

- `LOGGING['formatters']['context']`
- `LOGGING['formatters']['context_color']`
- `LOGGING['formatters']['logstash']`

Tuning the System

IN THIS SECTION

- CPU | 24
- Memory | 25
- Storage | 25
- Network | 26
- Apache | 27
- OpenVPN | 29
- PostgreSQL | 29
- Unattended Software Upgrades | 30
- Control Center | 30

This chapter contains recommendations on how to tune a Paragon Active Assurance system for optimum performance.

CPU

BIOS

Processors typically provide settings to enable, disable, and tune processor-level features. Today's systems usually provide maximum performance by default and do not need to be adjusted. However, we recommend that you ensure Turbo Boost is enabled in order to achieve slightly higher performance.

Scaling Governors

Linux supports different CPU scaling profiles (powersave, performance) that control the CPU clock frequencies via the kernel. Use the performance profile to achieve the best performance. To ensure that

the maximum clock frequency is always used, you need to set this for all CPUs. You can find out the number of CPUs with the `nproc` command. Then run the following for each CPU X:

```
echo performance > /sys/devices/system/cpu/cpufreq/policyX/scaling_governor
```

Note that setting the CPUs to run at maximum frequency is applicable to hardware systems only and may come with added costs.

Memory

The `swappiness` Linux kernel parameter controls how much (and how often) the Linux kernel will copy RAM contents to the swap space.

We recommend setting the amount of swapping to a minimum, without disabling it entirely, so that the memory pages are kept in physical memory:

```
sudo sysctl -w vm.swappiness=1
```

(The `sysctl` command is omitted from the kernel parameter settings that follow below.)

Storage

Partitions and File Systems

This is addressed in the Installation Guide, chapter [Installing Required OS and Software](#).

In addition, page cache flushing may be tuned to provide a more even behavior: background flush earlier, aggressive flush later.

```
vm.dirty_ratio = 80  
vm.dirty_background_ratio = 5  
vm.dirty_expire_centisecs = 12000
```

Read-ahead Size

The tunable `read_ahead_kb` parameter for storage block devices defines the maximum number of kilobytes that the operating system may read ahead during a sequential read operation. As a result, the likely-needed information is already present within the kernel page cache for the next sequential read, which improves read I/O performance.

We recommend setting this parameter to 4096 KB for all block devices:

```
echo 4096 > /sys/block/sdN/queue/read_ahead_kb
```

Network

Socket and TCP Buffers

The maximum socket buffer size (in bytes) should be set as follows for all protocol types and for both reads and writes to support full-duplex 10GbE connections:

```
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
```

To improve TCP throughput, set the auto-tuning parameters for the TCP read and write buffers: the minimum, default, and maximum number of bytes to use. We recommend the settings below. Increasing the minimum and default will consume more memory per connection, which may not be necessary.

```
net.ipv4.tcp_wmem = 4096 349520 16777216
net.ipv4.tcp_rmem = 4096 349520 16777216
```

TCP Backlog

The following TCP settings should be increased from their defaults to better handle bursts of load.

TCP backlog queue for half-open connections:

```
net.ipv4.tcp_max_syn_backlog = 8192
```

TCP backlog queue for passing connections to accept:

```
net.core.somaxconn = 4096
```

Device Backlog

The length of the network device backlog queue per CPU should be increased from the default:

```
net.core.netdev_max_backlog = 1000
```

Local Port Range

Extend the range of IP ports that are allowed for TCP and UDP connections to provide enough ephemeral ports for the anticipated workload:

```
net.ipv4.ip_local_port_range = 10240 65535
```

Apache

Performance

The following settings are recommended in `/etc/apache2/conf-available/performance.conf`:

```
HostnameLookups Off
KeepAlive On
KeepAliveTimeout 2
MaxKeepAliveRequests 600
Timeout 120
```

Event Multi-processing

The following settings are recommended in `/etc/apache2/mods-enabled/mpm_event.conf`:

```
<IfModule mpm_event_module>
    StartServers 3
    ServerLimit 8
    MinSpareThreads 8
    MaxSpareThreads 75
    ThreadLimit 64
    ThreadsPerChild 25
    MaxRequestWorkers 200
    MaxConnectionsPerChild 10000
</IfModule>
```

Security

The following settings are recommended in `/etc/apache2/conf-available/security.conf`:

```
ServerTokens Prod
ServerSignature Off
TraceEnable Off
```

Enabling the Configurations

Make sure that all Apache configurations above are saved, then enable them as follows:

```
a2enconf performance
a2enconf security
a2enmod mpm_event
systemctl reload apache2
```

OpenVPN

Transmit Queue Length

To improve OpenVPN performance and throughput, it is advisable to adjust the transmit queue length for the tun interface:

```
echo "txqueuelen 2000" >> /etc/openvpn/netrounds.conf
systemctl restart openvpn@netrounds.service
```

PostgreSQL

The following settings are recommended in `/etc/postgresql/10/main/postgresql.conf` for best performance:

```
max_connections = 100
effective_io_concurrency = 128 # between 64 and 256 based on performance of underlying storage
stack
default_statistics_target = 100
checkpoint_completion_target = 0.9
random_page_cost = 1.1

max_worker_processes = 8
max_parallel_workers_per_gather = 1
max_parallel_workers = 8

shared_buffers = X # set to 25% of available memory
effective_cache_size =
work_mem =
maintenance_work_mem =

temp_buffers = 16MB
wal_buffers = 16MB
wal_compression = on
```

Unattended Software Upgrades

To avoid automatically installing software packages, we advise you to disable the `unattended-upgrades` service or to remove it entirely from the system:

```
systemctl stop unattended-upgrades
systemctl disable unattended-upgrades
```

Control Center

Finally, a number of Control Center parameters can be tuned according to the available resources and the system load. These include:

- `/etc/netrounds/netrounds.conf`
 - `CALL_EXECUTER_MAX_CHILDREN`: This configures how many background jobs can run in parallel. The default is 20. Each test and monitor (not each task) will periodically run as a background job. If you have many tests or monitors, you might want to set this value to match the number of running tests and monitors to prevent those jobs from queuing up and delaying the collecting of data. The current queue length can be seen as the `scheduled_call_latency` parameter returned by the `ncc status` command.
- `/etc/apache2/sites-available/netrounds-ssl.conf`
 - `WSGIDaemonProcess netrounds ... processes`: This configures how many HTTP requests to the Control Center GUI can be handled at the same time. The default is 10. If you have many or slow requests, you might need to increase this number. If no worker is available to start processing a request, a 504 Gateway Timeout response will be returned.

Troubleshooting

More tips on troubleshooting generally are found in the Troubleshooting chapter of the Installation Guide.

To contact Juniper technical support, file a ticket at support.juniper.net/support/requesting-support. In your ticket, please include the output from the command

```
ncc generate-troubleshooting-report
```

Deleting Licenses from the System

If for some reason you need to wipe all existing licenses from your Paragon Active Assurance system and start over, you can do so as follows:

```
sudo rm /var/lib/netrounds/license/*.lic  
sudo systemctl restart netrounds-license-daemon
```

To activate new licenses, follow the procedure in the Installation Guide, chapter Getting Started with Paragon Active Assurance.

Deleting Paragon Active Assurance Accounts

How to create accounts in Paragon Active Assurance is dealt with in the Installation Guide, chapter Getting Started with Paragon Active Assurance.

To delete a Paragon Active Assurance account, use the following command.

NOTE: This command will permanently delete the specified account. You should back up your database and be sure what you are doing before using this command.

```
ncc account-delete --help # Read help first!  
ncc account-delete <short name of product account>
```

In order to delete anything, you must add the `--force` option to the command. Even with this option added, you will be asked to confirm your action. The confirmation step can be suppressed by adding the `--noinput` option.

Uninstalling Paragon Active Assurance

IN THIS SECTION

- [Optional Steps | 32](#)

Installation of Paragon Active Assurance is covered in the *Installation Guide*.

To uninstall the Control Center software, run the following command:

```
sudo apt-get remove "netrounds-*
```

Alternatively, in order to delete the Control Center packages from the instance, you can add the `--purge` flag:

```
sudo apt-get remove --purge "netrounds-*
```

This will not remove the data or the configuration, however. To remove all data, do the following:

```
sudo rm -rf /var/lib/netrounds
sudo -u postgres dropdb netrounds
sudo -u postgres dropdb paa-plugins
sudo -u postgres dropdb -h localhost -p 7432 paa-metrics
```

where the password for the last command (for TimescaleDB) is postgres.

Optional Steps

If you are not using PostgreSQL for anything else, you may want to uninstall the PostgreSQL database:

```
sudo apt-get remove --purge postgresql
```

Further optional cleanup tasks are as follows:

- Remove the Test Agent applications remaining in the static folder:

```
sudo rm -rf /usr/lib/python3.6/dist-packages/netrounds
```

- Remove logs from apache2:

```
sudo rm -rf /var/log/apache2/netrounds*
```

- Remove the Paragon Active Assurance site from apache2:

```
sudo rm -rf /var/lib/apache2/site/disabled_by_admin/netrounds*
```

- Remove available sites for Paragon Active Assurance:

```
sudo rm -rf /etc/apache2/sites-available/netrounds*
```

- Remove the OpenVPN environment:

```
sudo rm -rf /var/lib/openvpn/netrounds.env
```

- Remove Kafka logs:

```
sudo rm -rf /var/log/kafka/netrounds*
```

- Finally, remove these directories if they are present:

```
sudo rm -rf /usr/share/netrounds
sudo rm -rf /usr/share/mibs/netrounds
sudo rm -rf /usr/lib/netrounds
```

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners. Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice. Copyright © 2022 Juniper Networks, Inc. All rights reserved.