

Paragon Active Assurance Lifecycle Management Guide

Published
2021-08-19

RELEASE
3.0.0

Table of Contents

Updating the Database with a New User Password

Backing Up Product Data

Restoring Product Data from Backup

Monitoring the System

Managing Control Center and NETCONF Users

Importing and Exporting Test and Monitor Templates

Configuring the Lifetime of REST API Tokens

Tuning the System

Troubleshooting

Deleting Paragon Active Assurance Accounts

Deleting the MySQL Database

Uninstalling Paragon Active Assurance

Updating the Database with a New User Password

IN THIS SECTION

- Using special characters in the password | 2

The [Installation Guide](#) tells how to change the password for the "netrounds" user in the configuration files `/etc/netrounds/netrounds.conf` and `/etc/netrounds/probe-connect.conf`. After doing this, you must also 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: <https://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.

Backing Up Product Data

IN THIS SECTION

- [Backing Up the PostgreSQL Database | 3](#)
- [Backing Up the OpenVPN Keys | 4](#)
- [Backing Up the Configuration Files | 4](#)
- [Backing Up the RRD Files | 4](#)

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 systemctl stop "netrounds-*" apache2 kafka openvpn@netrounds
```

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

```
sudo systemctl start --all "netrounds-*" apache2 kafka openvpn@netrounds
```

Backing Up the PostgreSQL Database

Run this command:

```
pg_dump -h localhost -U netrounds netrounds > ncc_postgres.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: Netrounds versions before 2.29 used a MySQL database. If you are still using an old version, or if you are about to upgrade, use this command to back up a MySQL database:

```
mysqldump --user=root --password netrounds > ncc_mysql.sql
```

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 > ncc_postgres.binary
```

Backing Up the OpenVPN Keys

Use this command:

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

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/netrounds.conf
- /etc/netrounds/probe-connect.conf
- /etc/netrounds/restol.conf
- /etc/netrounds/secret_key
- /etc/netrounds/test-agent-gateway.yaml
- /etc/openvpn/netrounds.conf

For example:

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

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 ncc_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.

Restoring Product Data from Backup

Here is how to restore Paragon Active Assurance data from backup files. In this chapter, the backup files are assumed to be named in the same way as in the chapter ["Backing Up Product Data" on page 2](#).

NOTE: The procedure that follows requires that Control Center has been installed (so that the netrounds database exists).

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

```
sudo systemctl stop "netrounds-*" apache2 kafka openvpn@netrounds
```

2. **Drop the PostgreSQL database:**

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

3. **Recreate the database:**

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

4. **Restore the database:**

```
sudo -u postgres psql --set ON_ERROR_STOP=on netrounds < ncc_postgres.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 are restoring from a backup of an earlier version of Control Center than the one you currently have installed, you also need to run `sudo ncc migrate`.

NOTE: Netrounds versions prior to 2.29 used a MySQL database. If you are restoring such a database, please use this command instead:

```
mysql --user=root --database=netrounds --password < ncc_mysql.sql
```

If you backed up the database in binary format, use this command to restore it:

```
pg_restore -Fc -h localhost -d netrounds -c --verbose --disable-triggers
netrounds_db.bak -U netrounds --password
```

5. Restore the OpenVPN keys:

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

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

6. Restore RRD files, either from a tarball or from a snapshot. Compare the chapter "[Backing Up Product Data](#)" on [page 2](#). 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 ncc_rrd.tar.gz -C /
```

7. Copy the configuration files into their original locations.

8. **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](#).)

9. **Start** Paragon Active Assurance services:

```
sudo systemctl start --all "netrounds-*" apache2 kafka openvpn@netrounds
```

Monitoring the System

IN THIS SECTION

- [System Parameters | 7](#)
- [Application Parameters | 8](#)
- [Licenses | 8](#)
- [Processes | 8](#)
- [Logs | 9](#)

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%

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 time remaining of the validity period for the license activated on the server can be inspected with the command

```
ncc license show
```

In the output from this command, look for the following:

- `violated_after`: On this date, the license will expire. The system will still work, but a warning will be shown to users.
- `disabled_after`: On this date, the system will be disabled. Users will still be able to access the system to view results and reports, but no new measurements will be collected.

Processes

The following ncc processes should normally be running:

- netrounds-callexecuter
- netrounds-confd
- netrounds-probe-login
- apache2
- openvpn@netrounds

Logs

The file `/var/log/apache2/netrounds_error.log` by default contains all logs from Control Center. The configuration of the logging is done using the Python logging module configuration schema (<https://docs.python.org/2/library/logging.config.html#configuration-dictionary-schema>).

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']`

Managing Control Center and NETCONF Users

IN THIS SECTION

- Managing Control Center Users | 10

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 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 **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 | 12](#)
- [Exporting Templates | 13](#)
- [Importing Templates | 13](#)
- [Usage Examples | 13](#)

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
```

Configuring the Lifetime of REST API Tokens

The lifetime of REST API tokens is limited and is 10 years by default. This is governed by the parameter `REST_TOKEN_LIFETIME` in the file `/etc/netrounds/netrounds.conf`.

If you intend to use the REST API, you might need to change the value of this parameter to whatever is required in your case.

NOTE: In connection with an upgrade, you need to set the desired lifetime value for existing tokens prior to running the `ncc migrate` command.

Tuning the System

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.
- `/etc/postgresql/9.5/main/postgresql.conf`
 - `max_connections`: As a rule of thumb, this value should be roughly equal to `CALL_EXECUTER_MAX_CHILDREN + WSGIDaemonProcess processes + 50`.
 - `shared_buffers`: For advice on setting this parameter, see this page: <https://www.postgresql.org/docs/current/runtime-config-resource.html#GUC-SHARED-BUFFERS>.

REST API rate throttling can be applied in the file

- `/etc/netrounds/restol.conf`
 - The settings `RATE_LIMIT_ENABLED` and `RATE_LIMIT_DEFAULT` are used for throttling REST API requests. To enable throttling, set `RATE_LIMIT_ENABLED=True`.
 - Then configure the `RATE_LIMIT_DEFAULT` setting to indicate the maximum frequency of API requests from the same IP address. For example, `RATE_LIMIT_DEFAULT=30/second` will allow up to 30 requests per second from each IP address. The rate limit can be set per second, minute, hour, or day. It is also possible to set multiple limits, as in `RATE_LIMIT_DEFAULT=30/second,60/minute`.
 - To disable throttling, set `RATE_LIMIT_ENABLED=False`.

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 <https://support.juniper.net/support/requesting-support>. In your ticket, please include the output from the command

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

Deleting Paragon Active Assurance Accounts

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.

Deleting the MySQL Database

Netrounds versions before 2.29 used a MySQL database. In later versions a PostgreSQL database is used instead; however, on an installation originally using a MySQL database, MySQL will keep running following an upgrade. If there is a need to delete MySQL from the Control Center installation, use the following command:

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

Uninstalling Paragon Active Assurance

IN THIS SECTION

● [Optional steps | 17](#)

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 as follows:

```
sudo rm -rf /var/lib/netrounds
sudo -u postgres dropdb netrounds
```

NOTE: Versions before 2.29 used a MySQL database. To delete such a database:

```
mysql --user=root --password --execute="DROP DATABASE netrounds;"
```

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 clean-up tasks are as follows:

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

```
sudo rm -rf /usr/lib/python2.7/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
```