

Paragon Active Assurance Lifecycle Management Guide

Published
2021-08-10

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

Managing Keycloak

Importing and Exporting Test and Monitor Templates

Configuring the Lifetime of REST API Tokens

Tuning the System

Troubleshooting

Deleting Paragon Active Assurance Accounts

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 Licenses | 4](#)
- [Backing Up the Configuration Files | 4](#)
- [Backing Up the RRD Files | 5](#)

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: 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 Licenses

Use this command:

```
sudo tar -czf ncc_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/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`.

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 `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 **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
```

Managing Keycloak

IN THIS SECTION

- [What Is Keycloak? | 12](#)
- [Initial Keycloak Configuration | 13](#)
- [Advanced: Configuring Keycloak for LDAP or SSO | 13](#)
- [Configuring Control Center | 14](#)
- [Configuring Single Sign-On \(SSO\) | 14](#)
- [Podman Volume Mounts for Keycloak | 18](#)
- [Setting Up the Front-end URL for Keycloak | 18](#)
- [Managing Logging | 19](#)
- [Backup and Restore | 22](#)
- [Configuring the Database | 22](#)
- [Setting the Password Policy for the Realm | 23](#)
- [Setting Up SSL When Required for the Keycloak Instance | 24](#)
- [Control Center CLI for Keycloak Management | 25](#)

What Is Keycloak?

Keycloak is an open source software product which allows single sign-on (SSO) with identity and access management (IdM) aimed at modern applications and services. Keycloak is used in Paragon Active Assurance to manage the database of users and optionally to set up SSO.

NOTE: Keycloak has many settings and options available. It is highly recommended that you only configure the things mentioned explicitly in this chapter.

Initial Keycloak Configuration

Many aspects of the initial configuration of Keycloak are handled by a script which is run following installation of Paragon Active Assurance. You can proceed straight to the migrate command, which is done in the normal way with

```
ncc migrate
```

Please note that the migration takes considerable time; refer to the [Installation Guide](#), chapter "Installing Control Center and Related Tasks".

The above command does not migrate users and roles from Control Center into Keycloak. This must be done in a separate step:

```
ncc auth migrate --all
```

Advanced: Configuring Keycloak for LDAP or SSO

NOTE: This is required only when you want to configure LDAP or SSO; if neither of these is used in the Paragon Active Assurance setup, there is no need to create a Keycloak admin user.

By default no Keycloak admin user exists, and any external access to the admin page is restricted due to the sensitive nature of these settings.

To modify the Keycloak configuration, you need to create a Keycloak admin user and enable access to the Keycloak admin panel.

To create an admin user, run this command on your Control Center instance:

```
sudo ncc auth create-admin --keycloak-username admin --keycloak-password password
```

The admin panel is protected from external access. In order to access it, you need to use port forwarding when setting up an SSH connection:

The admin panel is protected from external access. In order to access it from a web client, you need to use port forwarding when setting up an SSH connection to your Control Center instance.

To do so, run the following command on the client computer where your web browser is located in order to open an SSH tunnel accessible on port 4200 (replace USER and HOSTNAME with the correct values):

```
ssh USER@HOSTNAME -L 4200:localhost:4200
```

This lets you access the admin panel from your browser by navigating to <http://localhost:4200/auth>.

If you are working in Windows, ssh is by default disabled. How to enable and use ssh commands in Windows is explained here: <https://www.howtogeek.com/336775/how-to-enable-and-use-windows-10s-built-in-ssh-commands>

How to Access the Admin Panel If You Have Forgotten the Admin Password

It is not possible to recover the administrator user password if it has been forgotten. In order to restore access to the admin panel, you need to create a new Keycloak admin user:

```
ncc auth create-admin --keycloak-username admin --keycloak-password password
```

This also restarts the Keycloak instance. The `ncc auth create-admin` command will wait until the instance is up and running. You can then log in with the new credentials and remove the old admin user.

Configuring Control Center

In order to properly configure Control Center to work in tandem with Keycloak, see the document *Upgrading Control Center to version 3.1*.

Configuring Single Sign-On (SSO)

Creating a Separate SSO Realm

Keycloak has a concept of *realms* to handle users, credentials, roles and groups. A user belongs to and logs in to a realm. Realms are isolated from one another and can only manage and authenticate the users that they control.

By default when Control Center is installed, a "Master" realm is created which stores the users in a local user database.

To set up SSO, you need to create a new realm that is connected to your Identity Provider (IdP). You create this separate SSO realm through the admin panel GUI:

- In the left-hand pane, below the current realm (by default "Master"), click the **Add realm** button.
- Type a name, then create the realm.

In the master realm, navigate to **Clients** and modify the automatically created `realm_name-realm` client:

- Change **Access Type** to "public".
- Enable **Direct Access Grants Enabled**.
- Disable **Standard Flow Enabled**.

Setting Up The New Realm and Client

In this section, `REALM_NAME` refers to the name you chose when creating the realm for SSO. In the commands that follow, replace `REALM_NAME` with that name.

When creating the SSO realm, Keycloak automatically created a new client in the master realm called `REALM_NAME-realm`. You should use this client for any administrative operations in the SSO realm.

NOTE: Clients are entities that can request Keycloak to authenticate a user. Most often, clients are applications and services that want to use Keycloak to secure themselves and provide a single sign-on solution. Clients can also be entities that just want to request identity information or an access token so that they can securely invoke other services on the network that are secured by Keycloak.

Run the following commands:

NOTE: Before you run the `setup-client` command, you must have set `SITE_URL` properly in `/etc/netrounds/netrounds.conf`.

```
export REALM_NAME="MySSOR realm" # Change this to your realm name
ncc auth setup-realm ${REALM_NAME}
```

This will configure the realm with proper password management settings, themes, SMTP settings, and more.

```
ncc auth setup-client account --realm ${REALM_NAME} --update-secret --secret SECRET
```

This will set up the client named account in the realm REALM_NAME with proper permissions and allow using it for login management.

```
ncc auth create CONFIG_NAME EMAIL_DOMAIN ${REALM_NAME} account SECRET -e CONTACT_EMAIL --is-active -d DESCRIPTION
```

Here,

- CONFIG_NAME is simply a name for the config in the database
- EMAIL_DOMAIN is the domain of the email addresses for which that particular config will be used (say, example.com)
- \${REALM_NAME} is the name of the realm this configuration will connect to. By default Keycloak creates only one realm, master, and it will be used most of the time.

Adding the Identity Provider

You need to add the REALM_NAME realm (referred to as the "SSO realm" below) as an identity provider for the Master realm used by Control Center.

1. Select the SSO realm at top left. In the left-hand pane, click **Identity Providers**, and select the SAML v2.0 protocol.
2. Either import the metadata from a file or URL, or fill in the **Single Sign-On Service URL** yourself and configure the **EntityID** to whatever your organization's identity provider requires.

NOTE: You need to obtain the metadata from the identity provider admin.

3. In this step we configure the identity provider further:

- Change **NameID Policy Format** to "email".
- Set **Validate Signature** ON.
- Set **Trust Email** ON.
- Click **Save**.

4. Add mappings for attributes that are named differently in the identity provider and Control Center.

- Click the **Mappers** tab for the identity provider.
- Add a mapper for `firstName` with the following settings:

Mapper Type	Attribute Importer
Sync Mode Override	force
Attribute Name	(identity provider specific)
Friendly Name	(identity provider specific)
User Attribute Name	firstName

- In the same way, add mappers for `lastName` and `username` with analogous settings.

5. You need to map roles in the SSO realm to roles in the Master realm. This step gives an example of such a mapping for the Control Center account "account1".

- In the SSO realm, click **Roles** in the left-hand pane.
- Click the **Add Role** button and create a role "paa-account1-admin". In the same way, create roles "paa-account1-read" and "paa-account1-write".
- Then navigate to **Identity providers** and select your identity provider. Click the **Mappers** tab.
- For each role, add a mapper of type "SAML Attribute to Role" with the following settings:

Sync Mode Override	inherit
Attribute Name	name of group or role list attribute
Attribute Value	name of group that should be mapped to the role
Role	name of role to map to, e.g. "paa-account1-admin"

- Click **Save**.

Setting Up Proper Redirection

This step needs to be done in the Master realm.

If you have configured SSO for a realm, only login using SSO should be allowed. To accomplish this, do the following for the realm that has SSO configured:

- Go to **Authentication > Flows > Browser**.
- Click **Actions > Config** for **Identity Provider Redirector**.
- Provide a name for the identity provider that provides SSO. This should be the alias of the identity provider added in the section ["Adding the Identity Provider" on page 16](#).
- Click **Save**.

Podman Volume Mounts for Keycloak

Keycloak runs as a Docker container using the Podman tool, which in turn is run as a SystemD service (refer to Podman documentation [here](#)). This means that all Keycloak software exists within that container, including logs, settings, and data. The following volumes are mounted from the host to the container:

- `/etc/netrounds/keycloak --> /opt/keycloak/standalone/configuration`
- `/var/log/keycloak --> /opt/keycloak/standalone/log`
- `/var/lib/netrounds/keycloak/data --> /opt/keycloak/standalone/data`

Setting Up the Front-end URL for Keycloak

By default Keycloak operates on the endpoint `https://HOSTNAME/auth`. This URL must be set correctly for the user to be redirected properly when trying to log in. If `HOSTNAME` changes for some reason (such as IP address change or domain setup), there are two ways to update Keycloak with the new information:

- Update the `/etc/hostname` file. It is then enough to restart the `netrounds-keycloak` service.

- Edit `/etc/netrounds/keycloak/standalone.xml`. Search for `frontendUrl` in the file, and in the following section set the value of the property `frontendUrl` to the proper value, for example, `https://NEW_HOSTNAME/auth`:

```
<provider name="default" enabled="true">
  <properties>
    <property name="frontendUrl" value="${keycloak.frontendUrl:}" />
    <property name="forceBackendUrlToFrontendUrl" value="false" />
  </properties>
</provider>
```

Restart the `netrounds-keycloak` service.

Managing Logging

The following section of the file `/etc/netrounds/keycloak/standalone.xml` is responsible for logging management:

```
<subsystem xmlns="urn:jboss:domain:logging:8.0">
  <console-handler name="CONSOLE">
    <level name="INFO" />
    <formatter>
      <named-formatter name="COLOR-PATTERN" />
    </formatter>
  </console-handler>
  <periodic-rotating-file-handler name="FILE" autoflush="true">
    <level name="INFO" />
    <formatter>
      <named-formatter name="JSON-FORMATTER" />
    </formatter>
    <file relative-to="jboss.server.log.dir" path="server.log" />
    <suffix value=".yyyy-MM-dd" />
    <append value="true" />
  </periodic-rotating-file-handler>
  <logger category="com.arjuna">
    <level name="WARN" />
  </logger>
  <logger category="io.jaegertracing.Configuration">
    <level name="WARN" />
  </logger>
</subsystem>
```

```

</logger>
<logger category="org.jboss.as.config">
  <level name="DEBUG"/>
</logger>
<logger category="sun.rmi">
  <level name="WARN"/>
</logger>
<root-logger>
  <level name="INFO"/>
  <handlers>
    <handler name="CONSOLE"/>
    <handler name="FILE"/>
  </handlers>
</root-logger>
<formatter name="PATTERN">
  <pattern-formatter pattern="%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c] (%t) %s%n"/>
</formatter>
<formatter name="COLOR-PATTERN">
  <pattern-formatter pattern="%K{level}%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%n"/>
</formatter>
<formatter name="JSON-FORMATTER">
  <json-formatter pretty-print="false">
    <exception-output-type value="formatted"/>
  </json-formatter>
</formatter>

```

By default all logs are stored in JSON format.

During the setup process we also enable storing of particular events in the database, along with all their useful information. These are not accessible to the ordinary user, as they do not have access to the Keycloak admin panel; however, exactly the same information appears in the logfiles which are stored in `/var/log/keycloak`. The image below shows the events configuration:

The screenshot displays the 'Events Config' interface. At the top, there are three tabs: 'LOGIN EVENTS', 'ADMIN EVENTS', and 'CONFIG'. The 'CONFIG' tab is active. Below the tabs, the 'Events Config' section shows 'Event Listeners' with a single entry: 'x jboss-logging'. The 'Login Events Settings' section includes a 'Save Events' toggle set to 'ON', a 'Saved Types' list containing 'x SEND_RESET_PASSWORD', 'x UPDATE_CONSENT_ERROR', 'x GRANT_CONSENT', and 'x REMOVE_TOTP', a 'Clear events' button, and an 'Expiration' field set to 'Hours'. The 'Admin Events Settings' section includes a 'Save Events' toggle set to 'ON', an 'Include Representation' toggle set to 'ON', a 'Clear admin events' button, and two buttons at the bottom: 'Clear changes' and 'Save'.

The full list of login and account events is as follows. Each event also has a corresponding error event.

Login Events

- Login: A user has logged in.
- Register: A user has registered.
- Logout: A user has logged out.
- Code to Token: An application/client has exchanged a code for a token.
- Refresh Token: An application/client has refreshed a token.

Account Events

- Social Link: An account has been linked to a social provider.

- Remove Social Link: A social provider has been removed from an account.
- Update Email: The email address for an account has changed.
- Update Profile: The profile for an account has changed.
- Send Password Reset: A password reset email has been sent.
- Update Password: The password for an account has changed.
- Update TOTP: The TOTP settings for an account have changed.
- Remove TOTP: TOTP has been removed from an account.
- Send Verify Email: An email verification email has been sent.
- Verify Email: The email address for an account has been verified.

Backup and Restore

See the chapters ["Backing Up Product Data" on page 2](#) and ["Restoring Product Data from Backup" on page 5](#).

Configuring the Database

By default Keycloak uses the same database system as Control Center. The Keycloak database is named keycloak, and the default user and password settings are keycloak and keycloak.

In order to change the keycloak user's password for the Keycloak PostgreSQL database, do the following:

- Run these commands:

```
sudo -u postgres psql keycloak
```

```
ALTER USER keycloak WITH PASSWORD 'new_password';
```


- Modify the `/etc/netrounds/keycloak/standalone.xml` configuration file. Find the following section and update the password node:

```
<datasource jndi-name="java:jboss/datasources/KeycloakDS" pool-name="KeycloakDS">
  <connection-url>jdbc:postgresql://postgres/keycloak</connection-url>
  <driver>postgres</driver>
  <pool>
    <min-pool-size>4</min-pool-size>
    <initial-pool-size>4</initial-pool-size>
    <max-pool-size>64</max-pool-size>
  </pool>
  <security>
    <user-name>netrounds</user-name>
    <password>new_password</password>
  </security>
</datasource>
```

- Restart Keycloak:

```
sudo service paa-keycloak restart
```

Setting the Password Policy for the Realm

By default during initial configuration of Paragon Active Assurance, a password policy will be created which is referred to as the "default" one in the Installation Guide, chapter *Password Strength in Paragon Active Assurance*. If the "strong" policy has been set in `/etc/netrounds/netrounds.conf`, that policy will be applied instead. In either case, the same password policy will be applied in both Paragon Active Assurance and Keycloak. This section tells you how to manually edit the password policy later on in Keycloak, should the need arise.

- Create a Keycloak admin user (if none yet exists) and log in to the Keycloak admin console. Follow the instructions in the ["prerequisites" on page 13](#) chapter.
- Navigate to **Authentication > Password Policy**.
- From the drop-down box select the *default* password policy and make the desired settings.
- Then select the *strong* password policy and make the desired settings.

The initial policy contains the regular expression

```
^(?!.*(\w)\1{1,}).+$
```

which prohibits multiple identical characters in succession: for example, "aa" or "abddd" is forbidden. On the other hand, "aba" is allowed.

Read more about password policies in Keycloak [here](#).

Setting Up SSL When Required for the Keycloak Instance

In order to change the SSL certificate and/or update the keystore password, do as follows:

- Remove old keystores (if they exist):

```
rm -f /etc/netrounds/keycloak/keycloak.jks
rm -f /etc/netrounds/keycloak/serverkeystore.p12
```

- Generate new keystores:

NOTE: What follows is an example of creating dummy certificates which are not secure. In a production system it is highly recommended that you create real SSL certificates using a certificate authority.

```
sudo openssl pkcs12 -export -name server -in /etc/ssl/certs/ssl-cert-snakeoil.pem -
inkey /etc/ssl/private/ssl-cert-snakeoil.key -out /etc/netrounds/keycloak/serverkeystore.p12 -
password pass:keycloak
sudo keytool -importkeystore -destkeystore /etc/netrounds/keycloak/keycloak.jks -
srckeystore /etc/netrounds/keycloak/serverkeystore.p12 -srcstoretype pkcs12 -alias server -
srcstorepass keycloak -deststorepass keycloak
```

- If you changed the password for the Keycloak keystore in the preceding commands, you need to update the file `/etc/netrounds/keycloak/standalone.xml` with the new password. Do so by changing the value of `keystore-password` in the following section:

```
<ssl>
  <keystore path="keycloak.jks" relative-to="jboss.server.config.dir" keystore-
```

```
password="secret" />
</ssl>
```

- Restart Keycloak:

Control Center CLI for Keycloak Management

Below is an overview of the Control Center CLI commands available for managing authentication via Keycloak. They all begin with `ncc auth`. Many of these have been used in various sections above.

```
usage: ncc auth [-h] [--version] [-v {0,1,2,3}] [--settings SETTINGS] [--pythonpath PYTHONPATH]
               [--traceback] [--no-color] {create-admin,init,status,migrate,create,update,reset,setup-
               realm,setup-client} ...
```

Positional arguments:

<code>create-admin</code>	Create an administrator user for Keycloak and restart it
<code>init</code>	Run initial configuration, create administrator user for Keycloak, set up realm and client using values from <code>netrounds.conf</code>
<code>status</code>	Status of integration between Keycloak and Control Center
<code>migrate</code>	Migrate data from Control Center into Keycloak. This is used when upgrading Control Center from an older version that did not use

Keycloak

<code>create</code>	Create an integration between Keycloak and Control Center
<code>update</code>	Update an integration between Keycloak and Control Center
<code>reset</code>	Reset an integration between Keycloak and Control Center
<code>setup-realm</code>	Set up realm integration between Keycloak and Control Center
<code>setup-client</code>	Set up client integration between Keycloak and Control Center

Optional arguments:

<code>-h, --help</code>	Show this help message and exit
<code>--version</code>	Show program version number and exit
<code>-v {0,1,2,3}, --verbosity {0,1,2,3}</code>	Verbosity level; 0 = minimal output, 1 = normal output, 2 = verbose output, 3 = very verbose output
<code>--settings SETTINGS</code>	The Python path to a settings module, e.g. <code>"myproject.settings.main"</code> . If this is not provided, the <code>DJANGO_SETTINGS_MODULE</code> environment variable will be used.

`--pythonpath PYTHONPATH`

Directory to add to the Python path, e.g. `"/home/djangoprojects/myproject"`.

<code>--traceback</code>	Raise on <code>CommandError</code> exceptions
<code>--no-color</code>	Do not colorize the command output

Importing and Exporting Test and Monitor Templates

IN THIS SECTION

- [Listing Templates | 26](#)
- [Exporting Templates | 27](#)
- [Importing Templates | 27](#)
- [Usage Examples | 27](#)

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/10/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.

Uninstalling Paragon Active Assurance

IN THIS SECTION

- [Optional steps | 31](#)

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
```

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
```