

Paragon Active Assurance Installation Guide

Published
2022-02-08

Table of Contents

Introduction

Prerequisites and Preparations

Installing Required OS and Software

Downloading Control Center and Test Agent Repository

Installing Control Center and Related Tasks

LDAP and TACACS+ Authentication

Getting Started with Paragon Active Assurance

Service Configuration

Password Strength in Paragon Active Assurance

Orchestration with NETCONF & YANG

Troubleshooting

Appendix: LDAP and TACACS+ Authentication

Introduction

Paragon Active Assurance Control Center can either be hosted by Juniper Networks or be installed at the customer site. This document describes how to perform an on-site installation of Control Center, to be operated and maintained by the customer's own staff.

Control Center is delivered as an application that is installed on an existing Ubuntu OS. (For precise OS requirements, see the section ["Operating System Requirements" on page 4.](#))

Prerequisites and Preparations

IN THIS SECTION

- [Hardware Requirements | 1](#)
- [Disk Usage | 2](#)
- [Required Communication Ports | 3](#)
- [Operating System Requirements | 4](#)

Hardware Requirements

- Control Center in production handling up to 2000 Test Agents and up to 20,000 active streams:
 - 8 vCPUs
 - 40 GB RAM
 - 2000 IOPS
 - 1 TB disk space
- Control Center in production handling up to 8000 Test Agents and up to 100,000 active streams:
 - 36 vCPUs

- 72 GB RAM
- 15,000 IOPS
- 5 TB disk space
- Lab trials of a Control Center with up to 20 Test Agents and up to 1000 active streams:
 - 2 vCPUs
 - 8 GB RAM
 - 100 IOPS
 - 10 GB disk space

Deployments larger than the limits indicated for each case will require multiple Control Centers.

Disk Usage

Below is some data on disk space taken up when testing various services. The total disk space required for a service is calculated by multiplying with the number of streams.

Service tested	Disk usage, standard resolution	Disk usage, high resolution
TWAMP	2.5 MB	180 MB
Bidirectional UDP	2 x 1.5 MB = 3.0 MB	2 x 110 MB = 220 MB

The measurement data resolutions "standard" and "high" are defined according to the tables that follow. As detailed below, each is composed of a set of progressively lower resolutions for progressively longer time spans. The total number of data points is that stored in the time series database for each metric from a stream for a time period of either one year (standard resolution) or five years (high resolution). When data points become older than that, they are purged from the database.

Standard resolution

Data points are stored for one year.

The total number of data points is 13,566 (the sum of all entries in the **Number of data points** column).

This is the default resolution in Paragon Active Assurance.

Time span	Data point resolution	Number of data points	Percentage of data points
-----------	-----------------------	-----------------------	---------------------------

Last 12 hours	10 s	4,320	31.8%
Last 2 days	1 min	2,880	21.2%
Last week	5 min	2,016	14.9%
Last month	20 min	2,160	15.9%
Last year	4 h	2,190	16.1%

High resolution

Data points are stored for five years.

The total number of data points is 963,385.

Time span	Data point resolution	Number of data points	Percentage of data points
Last 90 days	10 s	777,600	80.7%
Last 1.5 years	5 min	157,680	16.4%
Last 3 years	1 h	26,280	2.7%
Last 5 years	1 day = 24 h	1,825	0.2%

Required Communication Ports

When configuring a firewall, traffic on the following ports needs to be allowed to and from Control Center.

- Inbound:
 - TCP port 443 (HTTPS): Web interface
 - TCP port 80 (HTTP): Web interface (used by Speedtest, redirects other URLs to HTTPS)
 - TCP port 6000 (default): Encrypted OpenVPN connection for Test Agent Appliances
 - TCP port 6800: Encrypted WebSocket connection for Test Agent Applications

- Outbound:
 - TCP port 25 (SMTP): Mail delivery
 - UDP port 162 (SNMP): Sending SNMP traps for alarms
 - UDP port 123 (NTP): Time synchronization

In addition, the following ports need to be open on the `tun0` tunnel interface.

- Inbound:
 - TCP port 4334: General interaction with Test Agents
 - HTTP port 80: Test Agent firmware update requests
- Outbound:
 - TCP port 4334: General interaction with Test Agents

Operating System Requirements

- Ubuntu Server 18.04 LTS

A major release of Control Center will be "locked" to a version of the base OS, so a patch release of Control Center will not support a newer base OS than the original version.

Installing Required OS and Software

NOTE: Please note that this only describes a "fresh install". For upgrades, please refer to the Lifecycle Management document.

1. Install a clean Ubuntu 18.04 server.

- The system user name does not matter, *except* that the name "netrounds" is not allowed since PostgreSQL creates a user with that name (as described in ["this paragraph" on page 6](#)).
- Install only standard components (don't change the default selection).
- The following disk partitioning is recommended, especially for snapshot backups (but it is up to you as a user to decide):

- Recommended partitioning for lab setup:
 - /: Whole disk, ext4.
- Recommended partitioning for production setup:
 - /: 10% of disk space, ext4.
 - /var: 10% of disk space, ext4.
 - /var/lib/netrounds/rrd: 80% of disk space, ext4.
- No encryption
- Set the time zone to UTC, for example as follows:

```
sudo timedatectl set-timezone Etc/UTC
```

- Set all locales to en_US.UTF-8.
 - One way to do this is to manually edit the file /etc/default/locale. Example:

```
LANG=en_US.UTF-8
LC_ALL=en_US.UTF-8
LANGUAGE=en_US.UTF-8
```

- Make sure the following line is NOT commented out in the file /etc/locale.gen:

```
en_US.UTF-8 UTF-8
```

- Regenerate the locale files to make sure selected language is available:

```
sudo apt-get install locales
sudo locale-gen
```

2. Install NTP:

- First disable timedatectl:

```
sudo timedatectl set-ntp no
```

- Run this command:

```
timedatectl
```

and verify that

```
systemd-timesyncd.service active: no
```

- Now you can run the NTP installation:

```
sudo apt-get install ntp
```

- Make sure that the configured NTP servers are reachable:

```
ntpq -np
```

The output should normally be "all ones" expressed in octal.¹

3. Install PostgreSQL and set up a user for Control Center:

```
sudo apt-get update

sudo apt-get install postgresql

sudo -u postgres psql \
    -c "CREATE ROLE netrounds \           WITH ENCRYPTED PASSWORD 'netrounds' SUPERUSER LOGIN;"

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

Using an external PostgreSQL server is not recommended.

4. Install and configure an email server.

¹ In the output, the "reach" value for the NTP servers is an octal value indicating the outcome of the last eight NTP transactions. If all eight were successful, the value will be octal 377 (= binary 0b11111111). However, when you have just installed NTP, it is likely that fewer than eight NTP transactions have occurred, so that the value will be smaller: one of 1, 3, 7, 17, 37, 77, or 177 if all transactions were successful.

- Control Center will send emails to users:
 - when they are invited to an account,
 - when sending email alarms (i.e. if email rather than SNMP is used for this purpose), and
 - when sending periodic reports.
- Run the command

```
sudo apt-get install postfix
```

- For a simple setup where postfix can send directly to the destination email server, you can set **General type of mail configuration** to "Internet Site", and **System mail name** can usually be left as-is. Otherwise, postfix needs to be configured according to the environment. For guidance, refer to the official Ubuntu documentation at <https://help.ubuntu.com/lts/serverguide/postfix.html>.

Downloading Control Center and Test Agent Repository

These are the requisite files:

- Control Center tarball

```
paa-control-center_<version number>.tar.gz
```

- Test Agent repository

```
paa-test-agent_<TA Appliance ver. no.>_all.deb  
paa-test-agent-application_<TA Application ver. no.>.tar.gz
```

This software is available at <https://support.juniper.net/support/downloads>.

Also click the **Checksums** link for each file and note down the SHA256 checksum. These will be used to verify the integrity of the downloaded files (as detailed in the chapter "[Installing Control Center and Related Tasks](#)" on page 8).

Installing Control Center and Related Tasks

1. Install Control Center.

This procedure also installs the Paragon Active Assurance REST API.

```
export CC_VERSION=<enter version number here>

# Compute the checksum for the tar file and verify that it is equal to the SHA256
# checksum provided on the download page
sha256sum paa-control-center_${CC_VERSION}.tar.gz

# Unpack the tarball
tar -xzf paa-control-center_${CC_VERSION}.tar.gz

# Make sure packages are up to date
sudo apt-get update

# Start the installation
sudo apt-get install ./paa-control-center_${CC_VERSION}/*.deb
```

2. Run the database migration.

NOTE: This is a sensitive command, and care should be taken when executing it on a remote machine. In such a scenario it is strongly recommended that you use a program like `screen` or `tmux` so that the `migrate` command will continue running even if the `ssh` session breaks.

```
sudo ncc migrate
```

The `ncc migrate` command takes considerable time to execute (many minutes). It should print the following (details omitted below):

```
Migrating database...
Operations to perform:
  <...>
Synchronizing apps without migrations:
  <...>
```

```
Running migrations:
  <...>
Creating cache table...
<...>
Syncing test scripts...
<Updating script ...>
```

3. Install the Test Agent repositories:

```
export TA_APPLIANCE_VERSION=<enter version number here>
export TA_APPLICATION_VERSION=<enter version number here>

# Compute checksums for the repositories and verify that they match the
# SHA256 checksums provided on the download page
sha256sum paa-test-agent_${TA_APPLIANCE_VERSION}_all.deb
sha256sum paa-test-agent-application_${TA_APPLICATION_VERSION}.tar.gz

# Start the installation
sudo apt-get install \
./paa-test-agent_${TA_APPLIANCE_VERSION}_all.deb

sudo mkdir /usr/lib/python2.7/dist-packages/netrounds/static/test_agent/
sudo cp paa-test-agent-application_${TA_APPLICATION_VERSION}.tar.gz \
/usr/lib/python2.7/dist-packages/netrounds/static/test_agent/
```

4. Configure the REST API.

- Open the REST API configuration file:

```
sudo vim /etc/netrounds/restol.conf
```

In this file, set `SITE_URL` to the host name at which the HTTP clients should be able to access the REST API.

NOTE: It is crucial that this parameter be set correctly. For example, if that URL is <https://netrounds.local/rest/>, set

```
SITE_URL=netrounds.local
```

Regarding REST API rate limitation, see the [Lifecycle Management Guide](#). Control Center will automatically check for an Apache configuration for the REST API.

NOTE: By default, Control Center including the REST API uses snakeoil SSL certificates. It is strongly advised that you obtain real certificates and set `VERIFY_SSL=1` in the above config file. See also the section "[SSL Certificate Configuration](#)" on page 16.

LDAP and TACACS+ Authentication

See "[Appendix: LDAP and TACACS+ Authentication](#)" on page 24.

Getting Started with Paragon Active Assurance

1. **Create a user** in Control Center by running this command:

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

Example: `jane.doe@example.com`

2. **Create a superuser** which has access to all accounts in the system. Such a user must exist.

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

3. **Create an account** in Paragon Active Assurance with the user just created as owner:

```
ncc account-create --owner <email> --name "<full name of product account>" \
<short name of product account>
```

Example: Full name = "Example\ Corporation", short name = example. Note that spaces in the full name must be escaped.

NOTE: The account names are not the same as the owner's or some other user's personal name. The short account name will be used (for example) in the Control Center URL, while the full account name is what will normally be displayed to the user.

The `ncc account-create` command should normally be run only once, since by default you will have only one account in Paragon Active Assurance. The user specified as owner automatically gets admin permissions on the account.

4. Activate the Control Center license.

Control Center will not be fully operational until a valid license has been activated. Follow these steps to activate a license:

- Generate a UUID string with the command

```
ncc license license-request
```

- Log in to the Juniper EMS Portal at <https://license.juniper.net/licensemanage/> with the credentials you have received from Juniper.
- In the **My Product Licenses** view, click the **Activate** button for the relevant license.
- In the dialog that appears, under **SW Version**, leave the default choice **3.0 and Above**.
- Under **Universal Unique ID (UUID)**, enter the UUID string you generated in Control Center.
- Click the **Activate** button at the bottom of the screen.
- A license key will now be generated. Download it and save it as a plain-text file `cc_license.txt`.

NOTE: It is vital that you enter the UUID exactly as received from Juniper, using the standard UUID format in lowercase with hyphens: for example, `0a1b2c3d-4e5f-6789-a0b1-c2d3e4f5678`.

NOTE: If the license activation failed for some reason, for example because you mistyped the UUID, you can click the **Revoke** button in the **My Product Licenses** view to revoke the license. You then need to start over with the `ncc license license-request` command as described above.

- Now activate the license in Control Center using the command

```
ncc license activate cc_license.txt
```

This step is necessary in order to give the correct permissions to the account you just created. By default the account will be granted the full range of permissions granted by your license.

Usually you will have a single license file to activate. However, with Juniper's licensing model, you may in some cases obtain multiple license files. If so, you can either activate them one by one as described above, or you can concatenate them with

```
cat my-license-*.txt > all-licenses.txt
```

and then activate them all at once.

5. An account in Control Center is like a tenant that multiple users can have access to. **Creating additional users** can be done either from the command line or in the Control Center web GUI:

- To create new users from the command line, repeat the command

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

with different email addresses as desired.

To grant a user permissions on your Paragon Active Assurance account, use the command

```
ncc user-permission <email><short name of product account><permission>
```

where <permission> is one of "none", "read", "write", or "admin".

- To invite new users and grant them permissions via the Control Center web GUI, go to **Account > Permissions**.

For details on what a user is allowed to do at each permission level, see the in-app help under "Setting up your account" > "Administering users and permissions".

6. **Open the URL** `https://<host IP>/` in your browser and log in with the credentials specified when creating the user.

NOTE: The server uses a self-signed certificate by default. This will cause the web browser to issue a warning unless the certificate is replaced with a valid one.

7. **Download a Test Agent disk image** or OVA package.
 - See the in-app help under "Test Agents" > "Installing Test Agents".
8. **Install the Test Agent.**
 - See the in-app help under "Test Agents" > "Installing Test Agents".
9. **Specify the login server** as the Control Center host IP and a login server port of your choice. If you do not specify a port, the default port 6000 will be used.
 - For the general procedure, see the in-app help under "Test Agents" > "Configuring Test Agents from the local console" > "Configuring a Test Agent from the local console", subsection "Changing login server".
10. **Register the Test Agent.** Specify the server as <host IP>:<port>, where the IP and port are as defined in the preceding bullet. If no port is given, the default port 6000 is assumed.
 - For the general procedure, see the in-app help under "Test Agents" > "Configuring Test Agents from the local console" > "Registering a Test Agent from the local console".
11. **Run a ping test.**
 - See the in-app help under "Task types".
12. **Invite a user** and verify that the invite email is received.
 - See the in-app help under "Setting up your account" > "Administering users and permissions".

Service Configuration

IN THIS SECTION

- [Main Settings File | 14](#)
- [Apache | 15](#)
- [Test Agent Appliance Registration | 15](#)
- [SSL Certificate Configuration | 16](#)
- [PostgreSQL Configuration | 16](#)
- [How to Configure OpenVPN keys | 17](#)
- [HSTS Configuration | 17](#)

Services are configured by editing various configuration files, as detailed below. Go through this chapter and configure your settings as appropriate.

Summary of relevant configuration 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/test-agent-gateway.yaml`
- `/etc/openvpn/netrounds.conf`

Main Settings File

- `/etc/netrounds/netrounds.conf`

This file has inline documentation and examples for all supported settings. The `SITE_URL` setting is one that **always needs to be modified** to get the correct URL to Control Center, for example in emails and reports. It should be set the same as `SITE_URL` for the REST API (defined in `/etc/netrounds/netrounds.conf`).

Summary of settings in this file:

- Unique, secret string used for cryptographic operations
- Control Center web server URL
- Sender name in outgoing emails
- Contact email address shown to users
- Settings for sending email (backend, host, and more)
- PostgreSQL configuration (see the section ["PostgreSQL Configuration" on page 16](#))
- Logging configuration (for details see the section ["Logging" on page 20](#))
- Maximum length of log tags
- User session timeout (time of inactivity before the user is automatically logged out; governed by `SESSION_COOKIE_AGE`, set in seconds; one must also set `SESSION_SAVE_EVERY_REQUEST=True` in order for user actions to reset the timer)

- Number of days a password reset link is valid
- Criteria for automatic update of Test Agent software
- Storage location for time series data
- Storage location for OpenVPN certificates and keys used to authenticate Test Agents
- Number of tasks from the background task queue that can be processed in parallel
- Allow/Disallow sign-up for new account through the Control Center web GUI
- Send/Do not send notifications about SLA status changes to ConfD
- User time zone; the default is UTC

Apache

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

These files hold Apache settings.

For exhaustive information on this topic, please consult Apache documentation.

NOTE: It is strongly discouraged to change the Apache configuration files unless you are fully aware of the consequences. Inappropriate changes may break Paragon Active Assurance functionality.

Test Agent Appliance Registration

- `/etc/openvpn/netrounds.conf`

To configure the TCP port towards which Test Agent Appliances register, use the keyword `port` followed by the port number: for example, `port 6000`. The default port is 6000. Restart `openvpn@netrounds` for the configuration to take effect:

```
sudo systemctl restart openvpn@netrounds
```

SSL Certificate Configuration

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

This Apache configuration file contains the following SSL certificate settings, with default values as shown:

```
SSLCertificateFile      "/etc/ssl/certs/ssl-cert-snakeoil.pem"
SSLCertificateKeyFile   "/etc/ssl/private/ssl-cert-snakeoil.key"
```

For exhaustive information on this topic, please consult Apache documentation.

- `/etc/netrounds/test-agent-application.yaml`

This configuration file contains settings for the Test Agent Application gateway service (`netrounds-test-agent-gateway`), including SSL settings. It has the same snakeoil default certificates as above, which also should be changed.

- `/etc/netrounds/test-agent-gateway.yaml`

This configuration file contains SSL certificate settings for the Test Agent Application Gateway, which is used by Test Agent Applications to connect to Control Center.

```
# Test Agent Application config file
# Please run the command below to see available settings:
#   /usr/bin/test-agent-gateway-service --help
# SSL certificates used by the web server. Defaults to snakeoil.
ssl-cert: /etc/ssl/certs/ssl-cert-snakeoil.pem
ssl-key: /etc/ssl/private/ssl-cert-snakeoil.key
```

By default snakeoil SSL certificates are used in all cases, as seen in the code snippets above. These are created from the `ssl-cert` package which is preinstalled in Ubuntu. However, to ensure an encrypted and secure connection in a production environment, you are strongly advised to obtain proper, signed SSL certificates instead.

PostgreSQL Configuration

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

To configure the PostgreSQL server, you must edit settings in both of these files. Below is an example of file contents specifying database "database", user name "user", and password "password".

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

When updating the password in either of the configuration files, you need to escape any characters that are identical to the character surrounding strings (above, this is the single quote, '). The escape character used is the backslash.

When updating the password in the file `probe-connect.conf`, it is best to escape any non-alphanumeric characters. 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. How to do this is explained in the [Lifecycle Management Guide](#).

How to Configure OpenVPN keys

- `/etc/openvpn/netrounds.conf`

Here you configure the location of the OpenVPN keys. Restart OpenVPN for the configuration to take effect.

HSTS Configuration

- `/etc/netrounds/netrounds.conf`

HTTP Strict Transport Security (HSTS) is a web security policy mechanism which helps to protect websites against protocol downgrade attacks and cookie hijacking. It allows web servers to declare that web browsers (or other complying user agents) should only interact with it using secure HTTPS connections, and never via the insecure HTTP protocol.

A server implements an HSTS policy by supplying a header (Strict-Transport-Security) over an HTTPS connection. The header age is set to one hour.

NOTE: By default HSTS is disabled in Paragon Active Assurance since the Speedtest page uses HTTP for performance reasons. Uncomment the line below to enable HSTS if you are not using Speedtest.

```
# STRICT_TRANSPORT_SECURITY_HEADER = "max-age=3600; includeSubDomains"
```

Another way to allow enabling of HSTS in Control Center is to host Speedtest on a separate web server, as explained in the document [Creating a Custom Speedtest Web Page](#).

Password Strength in Paragon Active Assurance

The *default* password strength requirements for Control Center user passwords are as follows: Each password must contain

- at least 8 characters in total
- at least one digit
- at least one uppercase letter
- at least one lowercase letter.

It is possible to switch to a set of *stricter* requirements. According to these, each password must contain

- at least 12 characters in total
- at least one digit
- at least one uppercase letter
- at least one lowercase letter
- at least one special character

and must also not have any character occurring twice in a row.

To enforce the stricter requirements, make the following setting in the file `/etc/netrounds/netrounds.conf`:

```
USER_PASSWORD_STRENGTH='strong'
```

(You need to add the variable `USER_PASSWORD_STRENGTH` itself as it is by default not present in the configuration file.) This setting applies globally to all new users created in Control Center.

Orchestration with NETCONF & YANG

How to integrate Paragon Active Assurance with a network service orchestrator via the Control Center NETCONF/YANG API is described in the document [NETCONF & YANG API Orchestration Guide](#). That document also explains how to install ConfD, which is used as an intermediary between the Paragon Active Assurance system and NETCONF. Please note that ConfD is not included in the regular installation described in the present installation guide.

Troubleshooting

IN THIS SECTION

- [Logging | 20](#)
- [Root Shell Option in Test Agent Local Console | 20](#)
- [Problem: Services cannot be accessed | 21](#)
- [Problem: No data collected | 21](#)
- [Problem: Services fail | 22](#)
- [Problem: A Test Agent has successfully registered but does not come online \(status icon remains red\) | 23](#)
- [Contacting Juniper Technical Support | 24](#)

Logging

For troubleshooting it is generally helpful to check the following logs:

- `/var/log/apache/netrounds_access.log`: All HTTP requests made to the Control Center web interface.
- `/var/log/apache/netrounds_error.log`: 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.
- `/var/log/syslog`: General system log.
- `/var/log/mail.log`: Email server log.
- `dmesg`: Kernel log

It may also help to turn on additional logging in `/etc/netrounds/netrounds.conf` by changing the line

```
LOGGING['handlers']['console']['level'] = 'ERROR'
```

to

```
LOGGING['handlers']['console']['level'] = 'DEBUG'
```

Another way to show logging for the Paragon Active Assurance callexecuter is to use the `journalctl` utility:

```
journalctl -u netrounds-callexecuter
```

Read more about `journalctl` [here](#).

Root Shell Option in Test Agent Local Console

You can perform additional troubleshooting under the guidance of Paragon Active Assurance by logging into the root shell of a Test Agent. This is done from the Test Agent local admin console.

How to access the local console is explained in the in-app help under "Test Agents" > "Configuring Test Agents from the local console".

- Navigate to **Utilities** and select **Root shell**.

- Log in with password `onlyfordebugaccess`.

The root shell prompt now appears.

The root password can be changed within the root shell using the `passwd` command. Alternatively, this can be done using the **Change root password** option in the local console.



WARNING: Any changes made to the Test Agent in the root shell may cause the Test Agent to malfunction and/or void the warranty. When in doubt, always consult Juniper staff before proceeding.

The subsections that follow deal with specific problems.

Problem: Services cannot be accessed

Suggested actions:

- Check service status with the following commands:

```
sudo systemctl status "netrounds-*" openvpn@netrounds
sudo systemctl status apache2 openvpn@netrounds
```

- Check that the host name resolves to the correct IP, for instance using `dig` or `ping`
- Check listening ports with

```
netstat -lan
```

- Check network traffic with

```
tcpdump
```

Problem: No data collected

- *Possible cause:* NTP time sync problem. This will introduce a time offset in the server-generated result views, so that it may take a while until any results appear in these views.

- *Possible cause:* No free disk space.

Problem: Services fail

- *Possible cause:* Kafka has gone down.

If this happens, services relying on Kafka will also fail and try to reboot until Kafka comes back online.

You can check if Kafka is running with the following command:

```
sudo systemctl list-units --type=service | grep kafka
```

If Kafka has gone offline, then checking logs with

```
journalctl -u netrounds-test-agent-gateway.service
```

will return entries like the following:

```
Oct 17 11:17:05 example.com test-agent-gateway-service[23009]: {"level":"info","service":"test-agent-gateway-service","host":"example.com","src":"core","time":"2020-10-17T11:17:05.429473575Z","caller":"/app/cmd/server/"}
Oct 17 11:17:06 example.com systemd[1]: netrounds-test-agent-gateway.service: Main process exited, code=exited, status=1/FAILURE
Oct 17 11:17:06 example.com systemd[1]: netrounds-test-agent-gateway.service: Unit entered failed state.
Oct 17 11:17:06 example.com systemd[1]: netrounds-test-agent-gateway.service: Failed with result 'exit-code'.
Oct 17 11:17:07 example.com systemd[1]: netrounds-test-agent-gateway.service: Service hold-off time over, scheduling restart.
Oct 17 11:17:07 example.com systemd[1]: Stopped Netrounds TA3 Connection Service.
```


Problem: A Test Agent has successfully registered but does not come online (status icon remains red)

- *Possible cause:* The network does not allow the encrypted VPN connection to be established. Please check the configuration and logs of any firewall between the Test Agent and Control Center.

Detailed description:

The registration is done over HTTPS, while the connection setup after registration is done using OpenVPN on the same port. This can sometimes cause the network to allow the registration, but not the connection attempt.

- *Possible cause:* The clocks on the Test Agent and in Control Center are not in sync. Please check that both parties have NTP correctly configured, that their clocks are in sync with the NTP server, and that the NTP server clock in turn is also in sync.

Detailed description:

If the Test Agent clock is behind, then after the Test Agent registers, the TLS certificate signed by Control Center for the Test Agent will be invalid from the Test Agent's point of view until the Test Agent's clock has reached the time of signing according to the Control Center clock. Until that point, the Test Agent will not accept the certificate and will remain offline.

The TLS certificates are generated at the time of installing Control Center. If the clocks were not correct at this point, the certificates must be regenerated. Note that this also requires re-registration of all Test Agents. Follow these steps:

1. Make sure that the Control Center clock is in sync:

```
ntpq -np
```

2. Remove the old certificates:

```
rm /var/lib/netrounds/openvpn/*
```

3. Generate new certificates:

```
dpkg-reconfigure netrounds-probe-login
```

4. Register each Test Agent again under a different name.

Contacting Juniper Technical Support

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
```

Appendix: LDAP and TACACS+ Authentication

IN THIS SECTION

- [LDAP Authentication | 25](#)
- [TACACS+ Authentication | 30](#)

This appendix is relevant only if you wish to use an LDAP or TACACS+ server to authenticate Paragon Active Assurance users.

Paragon Active Assurance supports the use of LDAP or TACACS+ to manage and authenticate its users in a centralized way. The authentication is then done using a remote server instead of the local Control Center user database.

When a user attempts to log in to Control Center, the latter sends an authorization request to the LDAP or TACACS+ server. Based on the response, Control Center grants or denies the user access to Paragon Active Assurance accounts as detailed in the response.

The following notes apply to both LDAP and TACACS+ unless otherwise indicated. "Server" is used below to refer to either technology.

- If some account is defined twice in the mapping from LDAP/TACACS+ to Control Center, the user will receive the higher permission of the two granted. Thus, if the permission is set to "read" in one list element and to "admin" in another, the user will receive admin permission for that account.
- If one permission mapping grants permission to one account and another mapping denies it, then the user will receive access to the account.

- Account permissions are synchronized with the server on each user login. If the user is granted additional permissions by the Paragon Active Assurance local admin, these are valid only until the next time the user logs in. Conversely, if the user's privileges are changed on the server, these will not come into effect until next login.
- If the user name entered at login matches the email address of an existing Control Center user, the login will proceed using that user rather than a new one being created. However, during server authentication, all user profile details except the password will be overwritten with what is stored on the server, insofar as these details have been defined. The user can still log in using the existing Control Center password. This means that select users can have the password set in Control Center, so that it is still possible to log in if the LDAP/TACACS+ server goes down.
- If the user name entered at login does not exist in Control Center, the following happens:
 - For LDAP, the user's email address is read from the user `email` field. The name of this field can be changed using the setting `AUTH_LDAP_USER_ATTR_MAP`.
 - For TACACS+, no email field can be obtained from the server; instead, an attempt is made to parse the username entry as an email address.
 - If the email address is valid, it will be entered as email address in the Control Center database as well. However, the user will still have to log in with his LDAP/TACACS+ username.
 - If the email address is not valid, then:
 - For LDAP, an email address will be created with the structure `username@LDAP_EMAIL_DOMAIN` and entered into the database. Edit the settings file to change this domain.
 - For TACACS+ the same thing happens, but using the format `username@<TACACSPLUS_EMAIL_DOMAIN>`.

LDAP Authentication

We will illustrate this by reproducing a typical (OpenLDAP) server-side file with data preloaded into the LDAP database, and subsequently showing what corresponding configuration is necessary in Control Center.

Contents of `ldap.ldif`:

```
dn: dc=example,dc=com
objectClass: organization
objectClass: dcObject
objectClass: top
dc: example
o: example.com
```

```

dn: ou=users,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
cn: Users
member: uid=jsmith,dc=example,dc=com
member: uid=jane.smith@example.com,dc=example,dc=com
ou: users

```

```

dn: uid=jsmith,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: extensibleObject
cn: John Smith
givenName: John
sn: Smith
c: GB
telephoneNumber: +44 20 7946 0296
mail: john.smith@example.com
uid: jsmith
userPassword: Password1

```

```

dn: uid=jane.smith@example.com,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: extensibleObject
cn: Jane Smith
givenName: Jane
sn: Smith
c: GB
telephoneNumber: +44 20 7946 0580
mail: jane.smith@example.com
uid: jane.smith@example.com
userPassword: Password1

```

```

dn: ou=admins,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
cn: Administrators

```

```
member: uid=jdoe,dc=example,dc=com
member: uid=jane.doe@example.com,dc=example,dc=com
ou: admins
```

```
dn: uid=jdoe,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: extensibleObject
cn: John Doe
givenName: John
sn: Doe
c: GB
telephoneNumber: +44 20 7946 0344
mail: john.doe@example.com
uid: jdoe
userPassword: Password1
```

```
dn: uid=jane.doe@example.com,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: extensibleObject
cn: Jane Doe
givenName: Jane
sn: Doe
c: GB
telephoneNumber: +44 20 7946 0113
mail: jane.doe@example.com
uid: jane.doe@example.com
userPassword: Password1
```

This creates a total of four users, two with write privileges (jsmith and jane.smith@example.com) and two with admin privileges (jdoe and jane.doe@example.com). Note how two of the users have their user name as uid, while the other two have a uid consisting of an email address.

In order to enable LDAP authentication in Control Center, the following attributes have to be provided in the settings file `/etc/netrounds/netrounds.conf`:

```
# LDAP settings

import ldap
# NOTE: ActiveDirectoryGroupType is only needed if you're using Microsoft Active Directory.
from django_auth_ldap.config import LDAPSearch, GroupOfNamesType, ActiveDirectoryGroupType

AUTH_LDAP_ENABLED = True
AUTH_LDAP_START_TLS = True
AUTH_LDAP_SERVER_URI = 'ldap://ldap.example.com'
AUTH_LDAP_USER_DN_TEMPLATE = 'uid=%(user)s,dc=example,dc=com'

# LDAP user attribute value to use during lookup on database side
AUTH_LDAP_USER_QUERY_FIELD = 'email'

# This is the domain that will be appended to the username when the one from
# LDAP is not a valid email
AUTH_LDAP_EMAIL_DOMAIN='example.com'

# Set up the basic group parameters
# For vanilla LDAP such as OpenLDAP, use this.
GROUP_OBJECTCLASS_NAME = 'groupOfNames'
# Uncomment this if using Microsoft Active Directory
# GROUP_OBJECTCLASS_NAME = 'group'

AUTH_LDAP_GROUP_SEARCH = LDAPSearch('dc=example,dc=com',
                                     ldap.SCOPE_ONELEVEL,
                                     '(objectClass=%s)' % GROUP_OBJECTCLASS_NAME
                                    )

# LDAP group type used to indicate group membership for a user
AUTH_LDAP_GROUP_TYPE = GroupOfNamesType()
# Note: If you are using Microsoft Active Directory, use this instead:
# AUTH_LDAP_GROUP_TYPE = ActiveDirectoryGroupType()

# Mapping between LDAP groups and Control Center account/permission
AUTH_LDAP_ACCOUNT_GROUP_MAPPING = [
    {
        'dn': 'ou=users,dc=example,dc=com',
        'accounts': [
```

```

        {
            'name': 'dev',
            'permission': 'write'
        }
    ]
},
{
    'dn': 'ou=admins,dc=example,dc=com',
    'accounts': [
        {
            'name': 'dev',
            'permission': 'admin'
        }
    ]
}
]

# Populate the Django user from the LDAP directory
AUTH_LDAP_USER_ATTR_MAP = {
    'first_name': 'givenName',
    'last_name': 'sn',
    'email': 'mail',
    'phone': 'telephoneNumber',
    'country': 'c'
}

# LDAP connection options
AUTH_LDAP_CONNECTION_OPTIONS = {
    # Limit on waiting for a network response in seconds. This is a timeout
    # returned by poll/select following a connect in case of no activity.
    # OpenLDAP specific.
    ldap.OPT_NETWORK_TIMEOUT: 30,
    # Timeout value for the synchronous API calls in seconds. OpenLDAP specific.
    ldap.OPT_TIMEOUT: 30
}

# The following user must exist unless anonymous login is allowed
# Non-anonymous bind DN
AUTH_LDAP_BIND_DN = 'cn=admin,dc=example,dc=com'
# Non-anonymous bind password
AUTH_LDAP_BIND_PASSWORD = 'Password1'

# Finally, in order to get LDAP working, the first list item must be

```

```
# added below
AUTHENTICATION_BACKENDS = (
    'netrounds.domain.backends.LDAPAuthBackend',
    'netrounds.domain.backends.AuthBackend',
)
```

Most of the above follows what is documented at <https://django-auth-ldap.readthedocs.io/en/latest/reference.html#settings>.

TACACS+ Authentication

As for LDAP, we exhibit a typical server-side configuration file (for `tac_plus` from Pro-Bono-Publico: see http://www.pro-bono-publico.de/projects/tac_plus.html) and then show the necessary configuration in Control Center.

Contents of `config.cfg`:

```
id = spawnd {
    listen = { port = 49 }
}

id = tac_plus {
    host = any {
        key = "topsecret"
        address = 0.0.0.0/0
    }

    user = jbloggs {
        service = netrounds {
        }

        login = clear Password1
    }

    user = jane.bloggs@example.com {
        service = netrounds {
        }

        login = clear Password1
    }
}
```



```

user = jroe {
    service = netrounds {
        set permission = admin
    }

    login = clear Password1
}

user = jane.roe@example.com {
    service = netrounds {
        set permission = admin
    }

    login = clear Password1
}
}

```

This creates a total of four users, two of whom have admin privileges. Note how two of the users are identified by their user name, while the other two have user defined as an email address.

In order to enable TACACS+ authentication in Control Center, the following attributes need to be provided in the file `/etc/netrounds/netrounds.conf`:

```

# This is the TACACS+ server address
TACACSPLUS_HOST = 'tacacsplus.example.com'
TACACSPLUS_PORT = 49

# This is the secret shared by Control Center and TACACS+
TACACSPLUS_SECRET = 'topsecret'
TACACSPLUS_SESSION_TIMEOUT = 5
TACACSPLUS_AUTH_PROTOCOL = 'ascii'

# This is the domain that will be appended to the user name when the one from
# TACACS+ is not a valid email
TACACSPLUS_EMAIL_DOMAIN = 'example.com'

# Below, permissions from TACACS+ are mapped to accounts in Control Center
TACACSPLUS_SERVICE_TO_ACCOUNT_MAPPING = [
    {
        'arguments': [
            'service=netrounds'

```

```

    ],
    'accounts': [
        {
            'name': 'dev',
            'permission': 'write'
        }
    ]
},
{
    'arguments': [
        'service=netrounds',
        'permission=admin'
    ],

    'accounts': [
        {
            'name': 'dev',
            'permission': 'admin'
        }
    ]
}
]

# Finally, in order to get TACACS+ working, the first list item must be# added below
AUTHENTICATION_BACKENDS = (
    'netrounds.domain.backends.TACACSPlusBackend',
    'netrounds.domain.backends.AuthBackend',
)

```

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.