CARTA Controller

Release 3.0.0-beta.3

Angus Comrie, Adrianna Pińska and Robert Simmonds

Sep 30, 2022

CONTENTS:

CARTA is the Cube Analysis and Rendering Tool for Astronomy. This document describes the installation and configuration process for the controller component.

Detailed step-by-step instructions are provided for *Ubuntu 20.04 (Focal Fossa)* and *CentOS 8*. We officially support Ubuntu 18.04 and 20.04, and RHEL 7 and 8 (and their freely distributed binary-compatible alternatives, such as CentOS or AlmaLinux), with all available standard updates applied.

CHAPTER

ONE

INTRODUCTION

The CARTA controller provides a simple dashboard which authenticates users and allows them to manage their CARTA backend processes. It also serves static frontend code to clients, and dynamically redirects authenticated client connections to the appropriate backend processes. The controller can either handle authentication itself, or delegate it to an external OAuth2-based authentication server.

1.1 Dependencies

To allow the controller to serve CARTA sessions, you must give it access to an executable CARTA backend, which can be either a compiled executable or a container. If you want to use a non-standard version of the CARTA frontend, you must also build it, and adjust the controller configuration to point to it. You should use the `v3.0.0-beta.3` tag of the CARTA backend.

By default, the controller runs on port 8000. It should be run behind a proxy, so that it can be accessed via HTTP and HTTPS.

MongoDB is required for storing user preferences, layouts and (in the near future) controller metrics.

You also need a working NodeJS LTS installation with NPM. Use `npm install` to install all Node dependencies.

1.2 Authentication support

The CARTA controller supports four modes for authentication. All four modes use refresh and access tokens, as described in the OAuth2 Authorization flow, stored in JWT format. The modes are:

- **PAM authentication**: The PAM interface of the host system is used for user authentication. After the user's username and password configuration are validated by PAM, `carta-controller` returns a long-lived refresh token, signed with a private key, which can be exchanged by the CARTA dashboard or the CARTA frontend client for a short-lived access token.
- **LDAP authentication**: As above, but an LDAP server is used directly for user authentication.
- **Google authentication**: Google's authentication libraries are used for handling authentication. You must create a new web application in the Google API console. You will then use the client ID provided by this application in a number of places during the configuration.
- **External authentication**: This allows users to authenticate with some external OAuth2-based authentication system. This requires a fair amount of configuration, and has not been well-tested. It is assumed that the refresh token passed by the authentication system is stored as an `HttpOnly` cookie.

1.3 Getting help

If you encounter a problem with the controller or documentation, please submit an issue in the controller repo. If you need assistance in configuration or deployment, please contact the CARTA helpdesk.

1.4 Future work

Features still to be implemented:

- Better error feedback
- More flexibility with external auth

CHAPTER

TWO

INSTALLATION

2.1 Installing the backend

We provide binary Debian packages of the latest beta and release versions of the CARTA backend for Ubuntu 20.04 (Focal Fossa) and Ubuntu 18.04 (Bionic Beaver). You can install the beta version with all dependencies by adding our PPA to your system and running `apt-get install carta-backend-beta`. Please refer to our *Ubuntu Focal instructions* for more details.

Note: The `casacore-data` package is recommended by the backend package, but installing it is optional. The packages in our PPA should be compatible both with the `casacore-data` package in the core Ubuntu repositories and with the package provided by the Kern PPAs. You may also wish to manage the required data files without using a package.

To install the backend on a different host system, or to install a custom version, you can build it from source from the backend repository on GitHub.

2.2 Installing the frontend

If you install the controller from NPM, the corresponding packaged version of the frontend will also be installed automatically. If you wish to install the controller from source, or would like to use a custom frontend version, you can install it from the frontend repository on GitHub.

2.3 Installing the controller

You can install the beta version of the CARTA controller from NPM by running `npm install -g carta-controller@beta`, or from GitHub by cloning the controller repository and running `npm install`.

2.4 Running the controller

After you have installed the backend and the controller and edited the controller *configuration*, you can start the controller with `npm run start` (if installing from the source on GitHub) or just by running `carta-controller` (if installing the package from NPM). You can use a utility such as forever or pm2 to keep the controller running. It is also possible to create a pm2 startup script which will automatically start the controller when the system is rebooted.

CHAPTER

THREE

CONFIGURATION

3.1 System Configuration

3.1.1 CARTA backend permissions

The user under which the CARTA controller is running (assumed to be `carta`) must be given permission to use `sudo` to start `carta_backend` processes as any authenticated user and stop running `carta_backend` processes belonging to authenticated users. We provide a kill script which is only able to kill processes matching the name `carta_backend`. This makes it possible to restrict what processes the `carta` user is permitted to kill:

```
#!/bin/bash
# This script is a safe way to allow a user to kill specific commands of other users via␣
↪sudo
COMMAND_TO_MATCH="carta_backend"

# Gets the child PID of the command (because it is run via sudo)
CHILD_PID=`pgrep -P $1`
# Gets the command name of the process to be killed
COMMAND_OF_PID=`ps -p $CHILD_PID -o comm=`

# Only allow processes with the same command name to be killed
if [ "$COMMAND_OF_PID" == "$COMMAND_TO_MATCH" ]; then
    kill -9 $CHILD_PID
    exit $?
else
    echo "$COMMAND_OF_PID does not match $COMMAND_TO_MATCH"
    exit 1
fi
```

To provide the `carta` user with these privileges, you must make modifications to the sudoers configuration. An example sudoers config is provided. This example allows the `carta` user to run `carta_backend` only as users belonging to a specific group (assumed to be `carta-users`), in order to deny access to unauthorized accounts:

```
# customise this file to fit your environment using visudo /etc/sudoers.d/carta_
↪controller

# carta user can run the carta_backend command as any user in the carta-users group␣
↪without entering password
carta ALL=(%carta-users) NOPASSWD:SETENV: /usr/bin/carta_backend
```

(continues on next page)

(continued from previous page)

```
# carta user can run the kill script as any user in the carta-users group without␣
↪entering password
carta ALL=(%carta-users) NOPASSWD: /usr/local/bin/carta-kill-script
```

Warning: Please only edit your sudoers configuration with `visudo` or equivalent.

Note: Older versions of `sudo` do not support the `--preserve-env=VARIABLE` argument. If your version of `sudo` is too old, set `"preserveEnv"` to `false` in your controller configuration, and add `Defaults env_keep += "CARTA_AUTH_TOKEN"` to your sudoers configuration.

3.1.2 Authentication

When configured to use PAM or LDAP authentication, the controller signs and validates refresh and access tokens with SSL keys. You can generate a private/public key pair in PEM format using `openssl`:

```
cd /etc/carta
openssl genrsa -out carta_private.pem 4096
openssl rsa -in carta_private.pem -outform PEM -pubout -out carta_public.pem
```

PAM may be configured to use the host's local UNIX user authentication, or to communicate with a local or remote LDAP server. If the UNIX module is used for authentication, the `carta` user must be given read-only access to `/etc/shadow`. This is not required if you use PAM's LDAP module or the direct LDAP authentication method.

3.1.3 Nginx

We strongly suggest serving over HTTPS and redirecting HTTP traffic to HTTPS, especially if handling authentication internally. If you use nginx as a proxy, you can use this configuration example as a starting point to redirect incoming traffic from port 443 to port 8000:

```
server {
    listen 443 ssl;
    ssl on;
    server_name my-carta-server.com;
    ssl_certificate /etc/nginx/ssl/cert.pem;
    ssl_certificate_key /etc/nginx/ssl/key.pem;
    location / {
        proxy_set_header   X-Forwarded-For $remote_addr;
        proxy_pass http://localhost:8000/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}

server {
```

(continues on next page)

(continued from previous page)

```
    server_name my-carta-server.com;
    if ($host = my-carta-server.com) {
        return 301 https://$host$request_uri;
    }
    listen 80 ;
    listen [::]:80 ;
    return 404;
}
```

Please ensure that you include a trailing `/` when hosting the controller on a subdirectory (e.g. `location /carta/`).

You can also use other HTTP servers, such as Apache. Please ensure that they are set up to forward both standard HTTP requests and WebSocket traffic to the correct port.

3.1.4 Directories

By default, the controller attempts to write log files to the `/var/log/carta` directory. Please ensure that this directory exists and that the `carta` user has write permission.

3.2 Controller configuration

Controller configuration is handled by a configuration file in JSONC (JSON with JavaScript style comments) format, adhering to the *CARTA controller configuration schema*. An example controller configuration file is provided:

```
{
    "$schema": "./config_schema.json",
    "authProviders": {
        "pam": {
            "publicKeyLocation": "/etc/carta/carta_public.pem",
            "privateKeyLocation": "/etc/carta/carta_private.pem",
            "issuer": "my-carta-server.com"
        }
    },
    "database": {
        "uri": "mongodb://localhost:27017",
        "databaseName": "CARTA"
    },
    "serverPort": 8000,
    "serverInterface": "localhost",
    "processCommand": "/usr/bin/carta_backend",
    "killCommand": "/usr/local/bin/carta-kill-script",
    "rootFolderTemplate": "/home/{username}",
    "baseFolderTemplate": "/home/{username}",
    "dashboard": {
        "bannerColor": "#d2dce5",
        "backgroundColor": "#f6f8fa",
        "bannerImage": "/path/to/my/image.svg",
        "infoText": "Welcome to the CARTA server.",
        "loginText": "<span>Please enter your login credentials:</span>",
        "footerText": "<span>If you have any problems, comments or suggestions, please
```

(continues on next page)

(continued from previous page)

```
↪<a href='mailto:test@test.com'>contact us.</a></span>"
    }
}
```

By default, the controller assumes the config file is located at `/etc/carta/config.json`, but you can change this with the `--config` or `-c` command line argument when running the controller.

Configuration may also be added in separate files in a `config.d` directory in the same parent directory as the specified config file. Each file in this directory must be a valid configuration file. Any files found will be processed in alphabetical order, after the main configuration file.

The controller automatically executes the backend with the `--no_http` flag, to suppress the backend's built-in HTTP server. If the `logFileTemplate` configuration option is set, `--no_log` is also used to suppress user-level logs. `--port` is used to override the default port. `--top_level_folder` and a positional argument are used to set the top-level and starting data directories for the user, as specified in the `rootFolderTemplate` and `baseFolderTemplate` options, respectively.

To specify additional backend flags, we recommend editing a *global backend preferences* file. Most commandline arguments to the backend are also recognised as configuration options. The `additionalArgs` field in the controller configuration file can be used for any debug options which are not, and to disable the local or global configuration files.

If you use an external *authentication* system, you may need to translate a unique ID (such as email or username) from the authenticated external user information to an internal system user. You can do this by providing a user lookup table, which is watched by the controller and reloaded whenever it is updated:

```
# comment lines start with a #
user1@institute.ac user1
user2@institute.ac user2
# multiple usernames can be mapped to the same system user
user1_alt@trusted.domain user1
```

You can alter the controller's dashboard appearance by adjusting the `dashboard` field in the config file. You can change the banner image and background, and add login instructions or institutional notices.

The `httpOnly` flag can be used to disable secure signing of authentication tokens. This should only be used during initial deployment and testing, or debugging.

The controller assumes it is running at the root directory of your subdomain by default. If you would prefer to run on a subdirectory, you will need to specify the `dashboardAddress` and `apiAddress` values (relative to your subdomain) explicitly. For example, if you are hosting CARTA at `https://subdomain.domain.com/carta/version/v3-beta/`, you would need to include the following in your config file:

```
{
    "apiAddress": "/carta/version/v3-beta/api",
    "dashboardAddress": "/carta/version/v3-beta/dashboard"
}
```

3.3 Backend configuration

The global configuration file for the CARTA backend is located at `/etc/carta/backend.json`. A per-user configuration file can also be placed in each user's local CARTA preferences directory (typically `.carta` or `.carta-beta` in the user's home directory, depending on how the CARTA backend was installed). On a multi-user system, if users have write access to this location, you may wish to disable the use of per-user configuration files, to prevent users from bypassing the root directory configuration set by the controller. This must be done through the `additionalArgs` field in the *controller configuration*.

The backend configuration file must adhere to the *CARTA backend configuration schema*. An example backend configuration file is provided:

```
{
    "$schema": "https://cartavis.org/schemas/preference_backend_schema_2.json",
    "idle_timeout": 14400,
    "omp_threads": 8,
    "exit_timeout": 0,
    "initial_timeout": 30
}
```

3.4 Testing the configuration

To test the configuration of the controller, you can use the built-in test feature. Run `carta-controller --verbose --test <username>` as the `carta` user (or whichever user has the *added sudoers permissions*). `<username>` should be a user in the `carta-users` group. The expected output looks like this:

```
Checking config file /etc/carta/config.json
Adding additional config file config.d/pam.json
No top-level folder was specified. Reverting to default location
Testing configuration with user alice
Password for user alice:
✓ Checked PAM connection for user alice
✓ Verified uid (1000) for user alice
✓ Generated access token for user alice
✓ Checked database connection
✓ Checked log writing for user alice
✓ Read frontend index.html from /custom/frontend/path/build
[
'running sudo --preserve-env=CARTA_AUTH_TOKEN -n -u alice /usr/bin/carta_backend --no_
↪http --debug_no_auth --port 3499 --top_level_folder /usr/share/carta --no_log /usr/
↪share/carta'
]
[2021-11-30 12:28:48.207] [info] /usr/bin/carta_backend: Version 3.0.0-beta.3
[2021-11-30 12:28:48.209] [info] Listening on port 3499 with top level folder /usr/share/
↪carta, starting folder /usr/share/carta. The number of OpenMP worker threads will be␣
↪handled automatically.
✓ Backend process started successfully
[2021-11-30 12:28:50.169] [info] Session 1 [127.0.0.1] Connected. Num sessions: 1
✓ Backend process accepted connection
[ 'running sudo -u alice ./scripts/carta_kill_script.sh 54275' ]
✓ Backend process killed correctly
Controller tests with user alice succeeded
```

Note: If you run the controller from a source directory using `npm`, use `--` to ensure that any commandline parameters are passed to the controller and not to `npm`. For example: `npm run start -- --verbose --test alice`.

CHAPTER

FOUR

STEP-BY-STEP INSTRUCTIONS FOR UBUNTU 20.04.2 (FOCAL FOSSA)

Note: These instructions can be used almost unchanged on Ubuntu 18.04 (Bionic Badger). We note differences where they occur.

4.1 Dependencies

4.1.1 Install the CARTA backend and other required packages

```
# Add CARTA PPA
sudo add-apt-repository ppa:cartavis-team/carta
sudo apt-get update

# Install the beta backend package with all dependencies
sudo apt-get install carta-backend-beta

# Install additional packages
sudo apt-get install nginx g++ mongodb make curl
```

4.1.2 Set up directories and permissions

Ensure that all users who should have access to CARTA belong to a group that identifies them (assumed here to be called `carta-users`).

```
# create a 'carta' user to run the controller
sudo adduser --system --home /var/lib/carta --shell=/bin/bash --group carta

# add 'carta' user to the shadow group (only required for PAM UNIX authentication)
sudo usermod -a -G shadow carta

# log directory owned by carta
sudo mkdir -p /var/log/carta
sudo chown carta: /var/log/carta

# config directory owned by carta
```

(continues on next page)

(continued from previous page)

```
sudo mkdir -p /etc/carta
sudo chown carta: /etc/carta

# edit sudoers file to allow passwordless sudo execution of
# /usr/local/bin/carta-kill-script and /usr/bin/carta_backend
# by the carta user
sudo visudo -f /etc/sudoers.d/carta_controller
```

An *example sudoers configuration* is provided in the configuration section.

4.1.3 Configure nginx

A *sample configuration file* is provided in the configuration section. This should be adapted to your server configuration. The relevant part of the config is for forwarding / to port 8000.

4.2 Install CARTA controller

Note: Currently supported versions of NodeJS are v12, v14 and v16. In the example below we install the latest LTS version of NodeJS from the NodeSource repo. Do not pass the `--unsafe-perm` flag to `npm` if using a local install.

```
# Install the latest NodeJS LTS repo
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash -

# Install NodeJS, NPM and tools required to compile native addons
sudo apt-get install -y nodejs build-essential

# Install carta-controller (includes frontend config)
sudo npm install -g --unsafe-perm carta-controller@beta

# Install PM2 node service
sudo npm install -g pm2

# Switch to carta user
sudo su - carta

# Generate private/public keys
cd /etc/carta
openssl genrsa -out carta_private.pem 4096
openssl rsa -in carta_private.pem -outform PEM -pubout -out carta_public.pem
```

4.2.1 Configure controller

Edit `/etc/carta/config.json` to customise the appearance of the dashboard and other options. A *sample configuration file* is provided in the configuration section.

4.2.2 Run controller

This should be executed as the `carta` user.

```
pm2 start carta-controller
```

4.2.3 Create pm2 startup script

This service will start the controller automatically after a reboot. Please refer to the pm2 documentation for detailed instructions. You should run `pm2 startup` as `carta`, execute the generated command as a user with `sudo` access, and finally run `pm2 save` as `carta` to save the running controller process.

CHAPTER

FIVE

STEP-BY-STEP INSTRUCTIONS FOR CENTOS 8

Note: These instructions should also work for RHEL 8, AlmaLinux, and Rocky Linux. Some changes may be necessary for RHEL 7 / CentOS 7.

5.1 1. Install Node.js

The CARTA controller uses Node.js and supports v12, v14, and v16. Node.js can easily be installed from the CentOS 8 AppStream repository. Here we install v14, as well as the `npm` package manager.

```
# Install Node.js v14:
sudo dnf module enable nodejs:14
sudo dnf install -y nodejs npm

# Check it is installed and working:
node --version
npm --version
```

5.2 2. Install MongoDB

The CARTA controller uses MongoDB to store user preferences, etc.. MongoDB is not available through the default CentOS 8 repositories, but we can add a custom repository to install it more easily.

```
# Create a custom MongoDB repo file:
sudo cat <<EOT >> /etc/yum.repos.d/mongodb-org.repo
[mongodb-org-4.4]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/4.4/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.4.asc
EOT

# Install MongoDB:
sudo dnf update
sudo dnf install -y mongodb-org
```

(continues on next page)

(continued from previous page)

```
# Start and enable MongoDB to run on startup:
sudo systemctl start mongod
sudo systemctl enable mongod

# Check that it is working
sudo systemctl status mongod
```

Note: On RHEL7/CentOS7, MongoDB v14 can be installed as follows: `curl -fsSL https://rpm.nodesource.com/setup_14.x | bash - && yum install -y nodejs`

5.3 3. Install the CARTA controller

The easiest way to install the CARTA controller is using `npm`.

```
sudo dnf install -y python3 make gcc-c++
sudo npm install -g --unsafe-perm carta-controller@beta
```

Note: The CARTA controller executable will be installed at `/usr/local/lib/node_modules/carta-controller`. The CARTA frontend will be installed at `/usr/local/lib/node_modules/carta-controller/node_modules/carta-frontend/build`.

Note: Do not pass the `--unsafe-perm` flag to `npm` if using a local install.

Note: On RHEL7/CentOS7 the CARTA controller package can not run with the default gcc version 4.8.5 (there would be an error due to `node-linux-pam`). A work around is to install a newer GCC version from source in order to get a newer `libstdc++.so.6`, then add the location of the newer `libstdc++.so.6` to the LD_LIBRARY_PATH. After that, the CARTA controller can run on RHEL7/CentOS7.

5.4 4. Install the CARTA backend

The easiest way may be to install the CARTA backend is from our cartavis RPM repository.

```
# Install the CARTA backend
sudo dnf -y install 'dnf-command(copr)'
sudo dnf -y copr enable cartavis/carta
sudo dnf -y install epel-release
sudo dnf -y install carta-backend-beta

# Check that the backend can run and matches the major version number of the controller
/usr/bin/carta_backend --version
```

5.5 5. Install Nginx

The CARTA controller requires a webserver. Here we use NGINX, but Apache should work too.

```
# Install nginx:
sudo dnf install -y nginx
sudo systemctl start nginx
sudo systemctl enable nginx
sudo setsebool -P httpd_can_network_connect 1
sudo firewall-cmd --permanent --zone=public --add-service=http
sudo firewall-cmd --permanent --zone=public --add-service=https
sudo firewall-cmd --reload

# Set up the nginx configuration file using our sample configuration file linked below:
sudo cd /etc/nginx/conf.d/
sudo vi /etc/nginx/conf.d/carta.conf
sudo systemctl restart nginx

# Check it is running:
sudo systemctl status nginx
```

A *sample configuration file* is provided in the configuration section. This should be adapted to your server configuration.

Note: If there are problems, you can debug with `journactl -xe` and by checking log files in `/var/log/nginx/`.

5.6 6. Create the 'carta' user and modify sudoers

For security, we recommend not to run the CARTA controller as the root user. Therefore we create a new user called `carta`.

We will assign the group `carta-users` to every user account and enable them to run `/usr/bin/carta_backend` and the script to close the CARTA backend, `/usr/local/bin/carta-kill-script`, by adding a custom entry to the `sudoers` file.

```
# Create the carta user:
sudo adduser carta
# Check everything is OK
id carta
# It should show 'uid=1000(carta) gid=1000(carta) groups=1000(carta)'

# So that log files can be written:
sudo mkdir -p /var/log/carta
sudo chown -R carta /var/log/carta

# Add the custom sudoers file entry using our sample linked below
sudo visudo -f /etc/sudoers.d/carta_controller
```

An *example sudoers configuration* is provided in the configuration section.

Note: The only safe way to modify sudoers is using `visudo`. Any syntax errors from directly editing sudoers could

make your system unusable.

Note: The `carta` user should not be in the `carta-users` group. `carta-users` should only be assigned to the normal user accounts.

5.7 7. Set up the user authentication method

This is the most difficult step and depends on how you authenticate users at your institute. In this step-by-step guide we use PAM local authentication and a local user, `bob`, on the server running the CARTA controller. The user `bob` needs to be part of the `carta-users` group.

With PAM authentication, the `carta` user that runs the CARTA controller requires access to the `/etc/shadow` file in order to authenticate other users. We can enable this by creating a new group called `shadow` and assigning the `/etc/shadow` file to that group.

Note: Only PAM with local authentication requires `/etc/shadow` access. PAM using LDAP, and Google OAuth, do not require `/etc/shadow` access.

```
# Create the test user 'bob':
sudo useradd -G carta-users bob
sudo passwd bob

# A new group called 'shadow' needs to be assinged to the /etc/shadow file and user 'carta':
sudo groupadd shadow
sudo chgrp shadow /etc/shadow
sudo chmod g+r /etc/shadow
sudo usermod -a -G shadow carta
ls -l /etc/shadow
# It should show permissions as ----r-----. 1 root shadow
# It could be helpful to reboot the server at this point
sudo reboot
```

5.8 8. Configure the CARTA controller

Create and fill in the `config.json` using our *sample configuration file*. Also generate private/public keys as they are used by the CARTA controller to sign/verify/refresh access tokens.

```
sudo mkdir /etc/carta
sudo chown -R carta /etc/carta
vi /etc/carta/config.json

# Generate private/public keys:
cd /etc/carta
sudo openssl genrsa -out carta_private.pem 4096
sudo openssl rsa -in carta_private.pem -outform PEM -pubout -out carta_public.pem
```

Please check the CARTA Configuration Schema for all available options.

5.9 9. Check everything is working

Here we switch to the `carta` user and test the CARTA controller with our test user `bob`:

```
su - carta
carta-controller -verbose -test bob
```

If the test is successful, the CARTA controller should be ready to deploy.

5.10 10. Start the CARTA controller

```
su - carta
carta-controller
```

Now your users should be able to access your server's URL and log into CARTA.

5.11 Optional: Set up the CARTA controller to run with pm2

pm2 is a very convenient tool to keep the CARTA controller service running in the background, and even start it up automatically after a reboot.

```
sudo npm install -g pm2
su -carta
pm2 start carta-controller
```

Please refer to the pm2 documentation for detailed instructions.

CHAPTER

SIX

CARTA CONFIGURATION SCHEMA

Schema defining configuration options for the CARTA server

carta_config			
type	*object*		
properties			
• $schema	Reference to configuration schema file		
	type	*string*	
• authProviders	*AuthProviders*		
	Configuration option for authentication providers		
	type	*object*	
	default	pam	publicKeyLocation: /etc/carta/carta_public.pem privateKeyLocation: /etc/carta/carta_private.pem issuer: carta
	properties		
	• google	Google AuthProvider	
		Google AuthProvider	
	• pam	PAM AuthProvider	
		Local AuthProvider	
	• ldap	LDAP AuthProvider	
		LDAP AuthProvider	
	• external	External AuthProvider	
		External AuthProvider	
• database	Database configuration		
	type	*object*	
	default	uri	mongodb://localhost:27017
		databaseName	CARTA
	properties		
	• **uri**	MongoDB connection URI used to connect to a MongoDB deployment	
		type	*string*
		pattern	^mongodb://
		default	mongodb://localhost:27017
		format	uri
	• databaseName	Default database to connect to	
		type	*string*
		default	CARTA
	additionalProperties	False	
• serverPort	Port to listen on. It is advised to listen on a port other than 80 or 443, behind an SSL proxy		
	type	*integer* / *string*	
	examples	8000	

continues on next page

Table 1 – continued from previous page

		8080	
		/run/carta-controller	
		var/run/carta	
	maximum	65535	
	minimum	0	
	minLength	2	
	default	8000	
• serverInterface	Host interface to listen on. If empty, all interfaces are used		
	type	*string*	
	examples	localhost	
		127.0.0.1	
• httpOnly	Allow HTTP-only connections. For testing or internal networks only		
	type	*boolean*	
	default	False	
• serverAddress	Public-facing server address. If this is specified, all requests will be redirected to this address, otherwise any address used will be preserved		
	type	*string*	
	format	uri	
• dashboardAddress	Optional parameter for explicitly configuring the dashboard address. This can be absolute or relative. This is required if running the controller on a subdirectory		
	type	*string*	
	examples	https://my-server.com/carta/dashboard	
		/carta/dashboard	
		/carta-versions/dev/dashboard	
	format	uri-reference	
• apiAddress	Optional parameter for explicitly configuring a custom API base address. This can be absolute or relative. This is required if running the controller on a subdirectory		
	type	*string*	
	examples	https://my-server.com/carta/api	
		/carta/api	
		/carta-versions/dev/api	
	format	uri-reference	
• frontendPath	Path to the built frontend folder. If no path is provided, the packaged version will be used		
	type	*string*	
• backendPorts	Port range to use for the CARTA backend process		
	type	*object*	
	default	min	3003
		max	3500
	properties		
	• **min**	type	*integer*
		maximum	65535
		minimum	1024
	• **max**	type	*integer*
		maximum	65535
		minimum	1024
	additionalProperties	False	
• processCommand	Path to CARTA backend executable		
	type	*string*	
	examples	/usr/bin/carta_backend	
		/usr/local/bin/carta_backend	
	default	/usr/bin/carta_backend	
• preserveEnv	Use the –preserve-env argument when calling sudo		

continues on next page

Table 1 – continued from previous page

	type	*boolean*	
	default	True	
• killCommand	Path to CARTA kill script		
	type	*string*	
	examples	/usr/local/bin/carta-kill-script	
	default	/usr/local/bin/carta-kill-script	
• rootFolderTemplate	Top-level path of directories accessible to CARTA. The *{username}* placeholder will be replaced with the username. Defaults to */usr/share/carta* if it exists, or */usr/local/share/carta* if it exists. If neither exists and no default is provided, the controller exits with an error		
	type	*string*	
	examples	/home/{username}	
		/	
• baseFolderTemplate	Starting directory of CARTA. Must be a subfolder of rootFolderTemplate. The *{username}* placeholder will be replaced with the username. Defaults to the same value as *rootFolderTemplate*		
	type	*string*	
	examples	/home/{username}/CARTA	
		/data	
		/	
• logFileTemplate	Location of log file. The *{username}*, *{pid}* and *{datetime}* placeholders will be replaced with the username, process ID. and dat/time formatted as *YYYYMMDD.h_mm_ss* respectively		
	type	*string*	
	examples	/var/log/carta/{username}_{pid}.log	
		/home/{username}/CARTA/log/{datatime}_{pid}.log	
	default	/var/log/carta/{username}_{datetime}_{pid}.log	
• additionalArgs	Additional arguments to be passed to the backend process, defined as an array of strings. See backend documentation for details.		
	type	*array*	
	examples	–omp_threads	
		4	
		–initial_timeout	
		30	
		–exit_timeout	
		0	
	items	type	*string*
• startDelay	Wait time before checking whether started process is still running and sending a response to the connecting client		
	type	*integer*	
	minimum	0	
	default	250	
• dashboard	Dashboard appearance configuration		
	type	*object*	
	properties		
	• backgroundColor	Background color for the dashboard	
		type	*string*
		examples	red
			rgb(171 66 66)
			#ff11ee
		default	#f6f8fa
	• bannerColor	Background color for the institutional logo banner	
		type	*string*

continues on next page

Table 1 – continued from previous page

		examples	red
			rgb(171 66 66)
			#ff11ee
		default	#606f7e
	• bannerImage	Path to institutional logo in PNG or SVG format	
		type	*string*
	• infoText	Text displayed before and after sign in. Plain text or HTML	
		type	*string*
		examples	Welcome to the server
			Welcome to the server
	• loginText	Text displayed before sign-in only. Plain text or HTML	
		type	*string*
		examples	Please enter your username and password
			Click Sign in to log in via Google
	• footerText	Footer text. Plain text or HTML	
		type	*string*
		examples	Please contact the CARTA helpdesk for more information
			If you would like to access the server, or have any problems, comments or suggestions, please contact us.
• scriptingAccess	Control scripting access for users.		
	type	*string*	
	enum	enabled-all-users, disabled-all-users, opt-in	
	default	disabled-all-users	
additionalProperties	False		
if	properties		
	• serverPort	type	*string*
then	properties		
	• serverInterface	type	*null*

6.1 keyAlgorithm

Algorithm used for public/private keys

type	*string*
enum	HS256, HS384, HS512, RS256, RS384, RS512, ES256, ES384, ES512, PS256, PS384, PS512
default	RS256

6.2 Google AuthProvider

Authentication configuration when using Google authentication

type	*object*	
properties		
• **clientId**	Google application client ID	
	type	*string*
	examples	my-app-id.apps.googleusercontent.com
	pattern	^\S+.apps.googleusercontent.com$
• validDomain	Valid domains to accept. If this is empty or undefined, all domains are accepted. Domain specified by *hd* field in Google authentication configuration.	
	type	*string*
	examples	gmail.com
		my-google-domain.com
• useEmailAsId	Whether to use the email field as a unique identifier	
	type	*boolean*
	examples	True
		False
	default	True
• **userLookupTable**	Path of user lookup table as text file in format <unique user ID> <system user>. Example table given in *usertable.txt.stub*	
	type	*string*
	examples	/etc/carta/userlookup.txt
additionalProperties	False	

6.3 Local AuthProvider

Authentication configuration when using PAM-based authentication

type	*object*	
properties		
• **publicKeyLocation**	Path to public key (in PEM format) used for verifying JWTs	
	type	*string*
	examples	/etc/carta/carta_public.pem
• **privateKeyLocation**	Path to private key (in PEM format) used for signing JWTs	
	type	*string*
	examples	/etc/carta/carta_private.pem
• keyAlgorithm	default	RS256
	keyAlgorithm	
• **issuer**	Issuer field for JWT	
	type	*string*
	examples	my-carta-server
• refreshTokenAge	Lifetime of refresh tokens	
	type	*string*
	examples	1w
		15h

continues on next page

Table 2 – continued from previous page

		2d
	default	1w
• accessTokenAge	Lifetime of access tokens	
	type	*string*
	examples	90s
		1h
		15m
	default	15m
• scriptingTokenAge	Lifetime of scripting tokens	
	type	*string*
	examples	1w
		5d
		10h
	default	1w
additionalProperties	False	

6.4 LDAP AuthProvider

Authentication configuration when using LDAP-based authentication

type	*object*	
properties		
• **publicKeyLocation**	Path to public key (in PEM format) used for verifying JWTs	
	type	*string*
	examples	/etc/carta/carta_public.pem
• **privateKeyLocation**	Path to private key (in PEM format) used for signing JWTs	
	type	*string*
	examples	/etc/carta/carta_private.pem
• keyAlgorithm	default	RS256
	keyAlgorithm	
• **issuer**	Issuer field for JWT	
	type	*string*
	examples	my-carta-server
• refreshTokenAge	Lifetime of refresh tokens	
	type	*string*
	examples	1w
		15h
		2d
	default	1w
• accessTokenAge	Lifetime of access tokens	
	type	*string*
	examples	90s
		1h
		15m
	default	15m
• scriptingTokenAge	Lifetime of scripting tokens	
	type	*string*
	examples	1w
		5d
		10h

continues on next page

Table 3 – continued from previous page

	default	1w	
• **ldapOptions**	Options to path through to the LDAP auth instance		
	type	*object*	
	properties		
	• **url**	LDAP connection URI	
		type	*string*
		pattern	^ldaps?://
		format	uri
	• **searchBase**	Search base	
		type	*string*
	• searchFilter	Search filter to use	
		type	*string*
		default	uid={{username}}
	• starttls	Whether to start TLS when making a connection	
		type	*boolean*
		default	True
	• reconnect	Whether to automatically reconnect to LDAP	
		type	*boolean*
		default	True
	• bindProperty	Property of the LDAP user object to use when binding to verify the password	
		type	*string*
		default	dn
	• searchScope	Scope of the search	
		type	*string*
		enum	base, one, sub
		default	sub
	• bindDN	Admin connection DN, e.g. uid=myapp,ou=users,dc=example,dc=org. If not given at all, admin client is not bound.	
		type	*string*
	• bindCredentials	Password for bindDN	
		type	*string*
	• cache	If true, then up to 100 credentials at a time will be cached for 5 minutes	
		type	*boolean*
		default	False
	• strictDN	Force strict DN parsing for client methods	
		type	*boolean*
		default	True
	• idleTimeout	Milliseconds after last activity before client emits idle event	
		type	*number*
	additionalProperties	True	
additionalProperties	False		

6.5 External AuthProvider

OAuth2-compatible authentication configuration

type	*object*		
properties			
• **issuers**	List of valid issuers in JWT field		
	type	*array*	
	examples	my-auth-server	
		my-other-auth-server	
	items	type	*string*
	minItems	1	
• **publicKeyLocation**	Path to public key (in PEM format) used for verifying JWTs		
	type	*string*	
	examples	/etc/carta/my_auth_server_public_key.pem	
• keyAlgorithm	default	RS256	
	keyAlgorithm		
• **uniqueField**	Name of unique field to use as user ID		
	type	*string*	
	examples	user	
		sub	
		user_id	
• **tokenRefreshAddress**	Route for refreshing access tokens		
	type	*string*	
	pattern	^https?://	
	format	uri	
• logoutAddress	Route for logging out		
	type	*string*	
	pattern	^https?://	
	format	uri	
• userLookupTable	Path of user lookup table as text file in format <unique user ID> <system user>. If no user lookup is needed, this should be omitted. Example table given in *usertable.txt.stub*		
	type	*string*	
	examples	/etc/carta/userlookup.txt	
additionalProperties	False		

CHAPTER

SEVEN

BACKEND PREFERENCES

Schema for CARTA backend preferences (Version 2)

carta_backend_preferences_2		
properties		
• verbosity	type	*integer*
	enum	0, 1, 2, 3, 4, 5
	default	4
• no_log	type	*boolean*
	default	True
• log_performance	type	*boolean*
	default	False
• log_protocol_messages	type	*boolean*
	default	False
• no_frontend	type	*boolean*
	default	False
• no_database	type	*boolean*
	default	False
• no_http	type	*boolean*
	default	False
• no_browser	type	*boolean*
	default	False
• browser	type	*string*
	default	
• host	type	*string*
	minLength	1
	default	0.0.0.0
• port	type	*integer* / *array*
	default	3002
• omp_threads	type	*integer*
	default	-1
• top_level_folder	type	*string*
	minLength	1
	default	/
• frontend_folder	type	*string*
	minLength	1
	default	
• exit_timeout	type	*integer*
	default	-1
• initial_timeout	type	*integer*

continues on next page

Table 1 – continued from previous page

	default	-1
• idle_timeout	type	*integer*
	default	-1
• read_only_mode	type	*boolean*
	default	False
• starting_folder	type	*string*
	minLength	1
	default	
• event_thread_count	type	*integer*
	default	-1
• enable_scripting	type	*boolean*
	default	False