Rock Documentation

OBiBa

Mar 31, 2024

CONTENTS

1 Introduction **3**
- 1.1 Architecture . . . 3
- 1.2 Stateful R Sessions . . . 4
- 1.3 Security . . . 4
- 1.4 Scalability . . . 4
- 1.5 Distributed Computing . . . 4
- 1.6 Service Registration . . . 6

2 Installation **9**
- 2.1 Requirements . . . 9
- 2.2 Install . . . 9
- 2.3 Upgrade . . . 12
- 2.4 Execution . . . 12

3 Configuration **15**
- 3.1 HTTP Server Configuration . . . 15
- 3.2 Cluster Node Configuration . . . 15
- 3.3 R Server Configuration . . . 16
- 3.4 Users Configuration . . . 16
- 3.5 AppArmor Configuration . . . 17
- 3.6 R Configuration . . . 17

4 REST API Introduction **19**
- 4.1 Authentication . . . 19
- 4.2 Authorization . . . 19
- 4.3 Clients . . . 20

5 Service **21**
- 5.1 Information . . . 21
- 5.2 Check . . . 22

6 R Server **23**
- 6.1 Status . . . 23
- 6.2 Start . . . 24
- 6.3 Stop . . . 25
- 6.4 Log . . . 26

7 R Server Packages **27**
- 7.1 List . . . 27
- 7.2 Update . . . 29
- 7.3 Remove . . . 30

7.4 Install . . . 30
7.5 DataSHIELD . . . 31

8 R Server Package **35**
8.1 Description . . . 35
8.2 Remove . . . 37

9 R Sessions **39**
9.1 List . . . 39
9.2 Remove . . . 40
9.3 Create . . . 41

10 R Session **43**
10.1 Status . . . 43
10.2 Remove . . . 44
10.3 Assign . . . 45
10.4 Evaluate . . . 46
10.5 Files . . . 47
10.6 Commands . . . 50

11 Partners and Funders **57**

12 Support **59**

HTTP Routing Table **61**

Rock is the OBiBa's R server which intends to be easy to install and to use. Rock exposes a REST API which can be used by any client application with HTTP communication capability. The R server engine is based on Rserve that Rock uses to manage stateful R sessions.

Targeted at individual studies and study consortia, OBiBa software stack (Opal, Mica etc.) provides a software solution for epidemiological data management, analysis and publication. While Opal, the core data warehouse application, provides all the necessary tools to import, transform, describe and analyze data, Mica provides everything needed to build personalized web data portals and publish content of research activities of both studies and consortia.

Opal uses Rock to constitute clusters of R servers and to dispatch the analysis requests on multiple hosts.

CHAPTER

ONE

INTRODUCTION

Rock is a server application that provides a REST API to execute R operations on server side in the context of a stateful R session. Rock can be discovered or can self-register to build clusters for performing distributed computations.

1.1 Architecture

The Rock frontend is a REST API that is implemented by a Java application. This application manages in the backend, a R server process (based on Rserve), creates R sessions and controls the lifecycle and the access to these R sessions. System status is reported to allow service registration and load balancing (see *R Server* API documentation). This allows to constitute clusters of Rock servers.

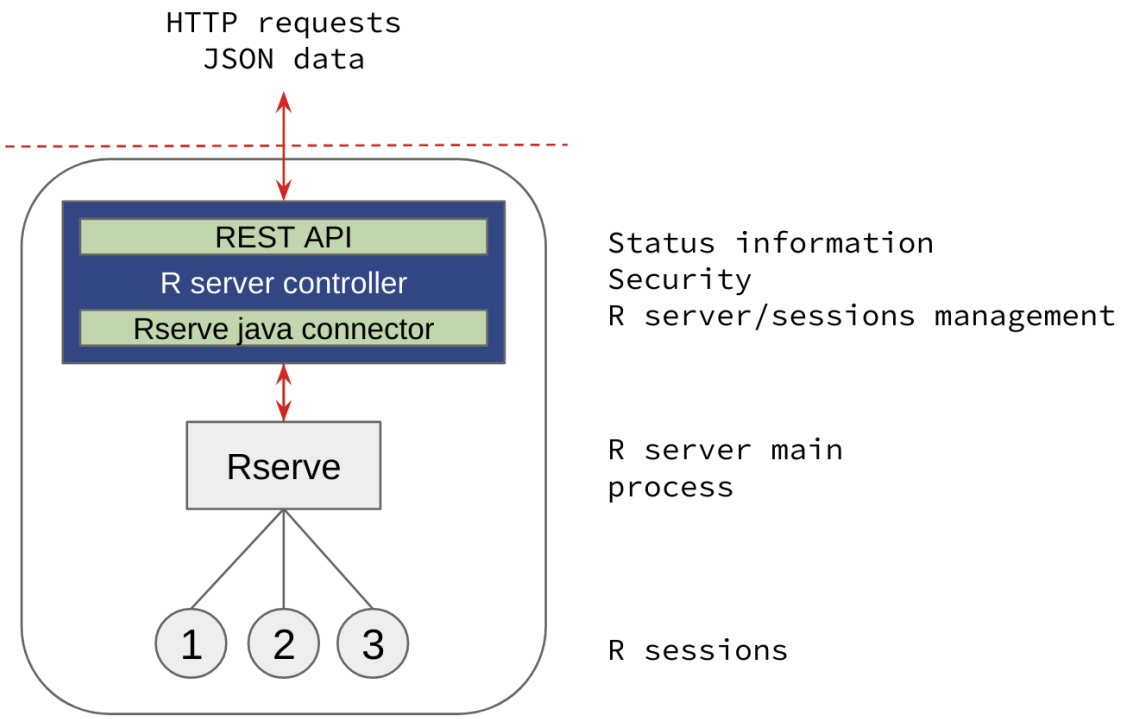

Fig. 1: Rock architecture combines Java (frontend) and R (backend) applications, to build a secure and easy to use service.

1.2 Stateful R Sessions

Unlike OpenCPU, the R sessions are stateful and then the interactions with the remote R session is very similar to the ones with a local R session. The main operations that can be performed on a remote R session are: assign, evaluate, upload/download file. See *R Session* entry points documentation.

The consequence of this is that the hardware resources (memory, CPU) are consumed on a single R server throughout the R session life: this makes it potentially less robust in case of the R server is terminated abruptly. Another aspect is that R objects living in the remote R session memory are immediately available for computation, but can affect the overall performance of the server when the memory is not managed correctly by the client application. In the end it is a trade-off between the ease of use and the reliability. Rock is more suitable as an application's backend service (such as Opal) than as a direct and general purpose R service (as proposed by OpenCPU).

1.3 Security

Rock requires user authentication and authorization. The user registry is file-based.

The following user roles are defined:

- `administrator`, can do anything.
- `manager`, can only manage the R server: R process and R packages. Creating and using R sessions is not permitted.
- `user`, can only create and use its own R sessions. R operations may be limited by the *AppArmor Configuration*.

A *user* is not necessarily a human being, it can also be an application using Rock as a computation engine.

1.4 Scalability

When the computation needs grow, horizontal scalability consists of expanding the number of Rock servers, whereas vertical scalability consists of expanding the hardware resources on a single Rock server.

In order to balance the load appropriately, Rock allows to group server instances (using *tags* information) and provides detailed information about the R server status, like the total number of R sessions, the busy ones, the number of system cores (to reduce the concurrency between R sessions) and the amount of available free memory. More information could be added in the future, to allow more refined load balancing strategies. See *R Server* Status API documentation.

1.5 Distributed Computing

As Rock offers the possibility to submit assignment and evaluation R operations in an asynchronous manner, a client application could use several Rock servers in parallel for dispatching the computation load on multiple hardware resources. Once all the managed R operations are completed, the results from each computed parts can be retrieved and combined. It would be fairly easy to implement such distributed computing process using the rockr R package for instance. See *Commands* API documentation.

One example of advanced distributed computing framework is DataSHIELD, which is based on Opal and offers privacy-preserving features.

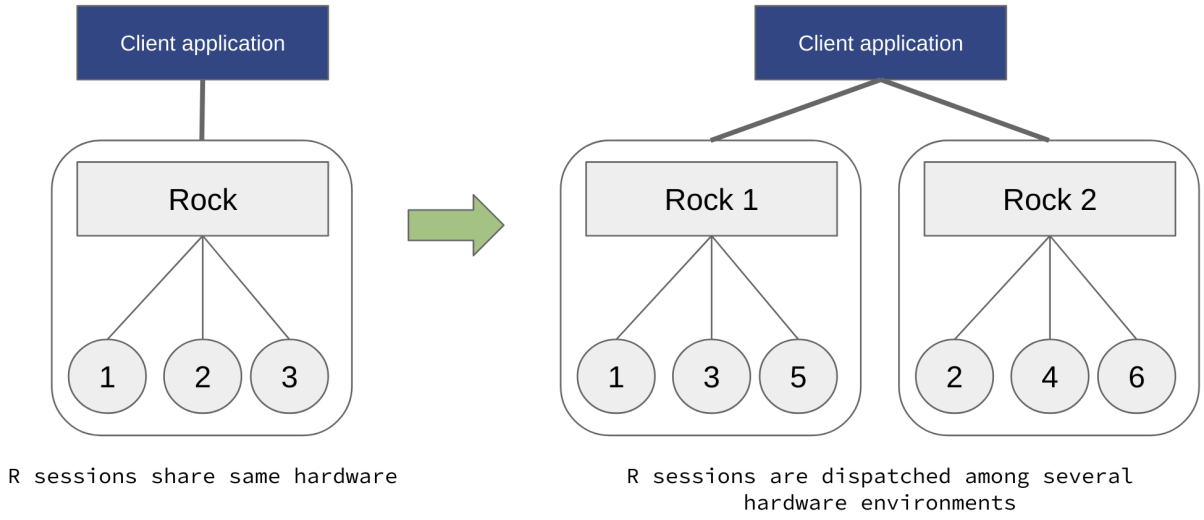

Fig. 2: Add more Rock servers and balance the load to scale R services horizontally.

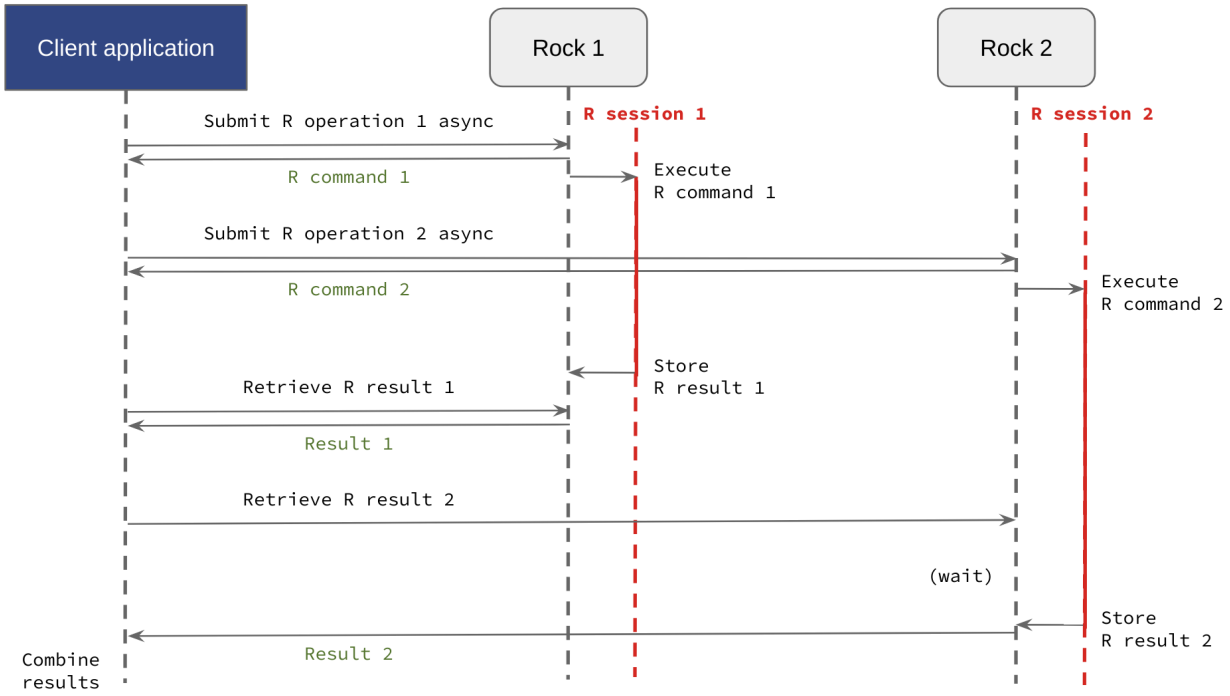

Fig. 3: R computations can be run in parallel and results be retrieved in a subsequent request, to be combined.

1.6 Service Registration

When used as an application's backend service, Rock can be used directly or published in a registry.

1.6.1 Discovery

When discovering the Rock service, the application with registry knows the location (base URL) of the Rock servers. When connection is established by the application (requires credentials), the *Service* entry points allow the application to get node information and perform periodical availability checks.

For example, Opal application is able to discover Rock servers from their base URL and do automatic registration and unregistration of the services.

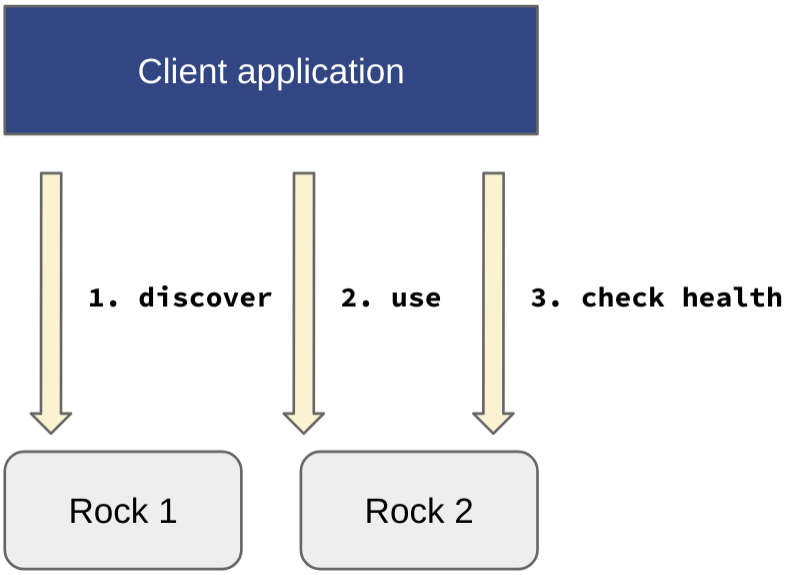

Fig. 4: Rock service discovery, use and health check.

1.6.2 Self-registration

In the self-registration process, Rock knows the location (base URL) of the registry. When connection is established by the Rock server (usually requires credentials), the *Service* entry points allow the registry to get node information and perform periodical availability checks.

Currently self-registration is proposed for Consul (see *Consul Configuration*) and Opal (see *Opal Configuration*).

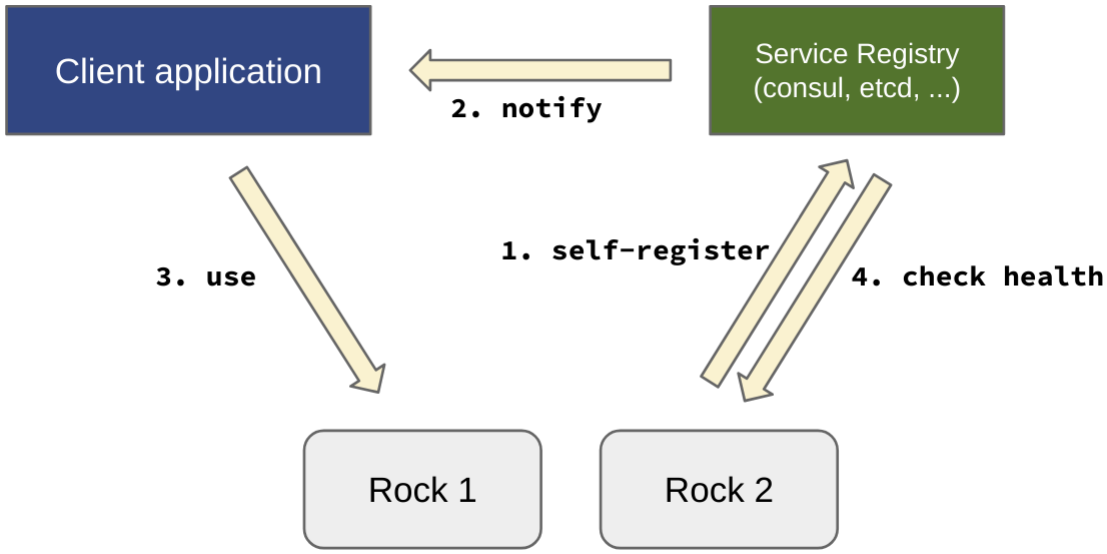

Fig. 5: Rock self-registration to a service registry.

CHAPTER

TWO

INSTALLATION

Rock is a stand-alone Java server application. The R computation engine is handled by Rserve and therefore requires a working R environment.

2.1 Requirements

2.1.1 Server Hardware Requirements

Component	Requirement
CPU	Recent server-grade or high-end consumer-grade processor
Disk space	8GB or more.
Memory (RAM)	Minimum: 4GB, Recommended: >4GB

2.1.2 Server Software Requirements

Software	Version	Download link	Usage
Java	21	OpenJDK downloads	Java runtime environment
R	any version	R project	R server engine

Both Java and R must be installed on the same host.

2.2 Install

Rock is distributed as a Debian/RPM package, as a zip file and as a Docker image. The resulting installation has default configuration that makes Rock ready to be used (as soon as R is available). Once installation is done, see *Configuration* instructions.

2.2.1 Debian Package Installation

Rock is available as a Debian package from OBiBa Debian repository. To proceed installation, do as follows:

- Install Debian package. Follow the instructions in the repository main page for installing Rock.
- Manage Rock Service: after package installation, Rock server is running: see how to manage the Service.

2.2.2 RPM Package Installation

Rock is available as a RPM package from OBiBa RPM repository. To proceed installation, do as follows:

- Install RPM package. Follow the instructions in the RPM repository main page for installing Rock.
- Manage Rock Service: after package installation, Rock is running: see how to manage the Service.

2.2.3 Zip Distribution Installation

Rock is also available as a Zip file. To install Rock zip distribution, proceed as follows:

- Download Rock distribution
- Unzip the Rock distribution. Note that the zip file contains a root directory named **rock-x.y.z-dist** (where x, y and z are the major, minor and micro releases, respectively). You can copy it wherever you want. You can also rename it.
- Create an `ROCK_HOME` environment variable
- Separate Rock home from Rock distribution directories (recommended). This will facilitate subsequent upgrades.

Set-up example for Linux:

```
mkdir rock-home
cp -r rock-x-dist/conf rock-home
export ROCK_HOME=`pwd`/rock-home
./rock-x-dist/bin/rock
```

Launch Rock. This step will create/update the database schema for Rock and will start Rock: see Regular Command.

For the administrator accounts, the credentials are "administrator" as username and "password" as password. See User Directories Configuration to change it.

2.2.4 Docker Image Installation

OBiBa is an early adopter of the Docker technology, providing its own images from the Docker Hub repository.

See also the Rock docker images that are DataSHIELD ready: datashield/rock-base

A typical docker-compose file would be:

```
version: '3'
services:
  rock:
    image: obiba/rock:latest
    ports:
      - ${PORT}:8085
    environment:
```

(continues on next page)

(continued from previous page)

```
    - ROCK_ADMINISTRATOR_NAME=${ROCK_ADMINISTRATOR_NAME}
    - ROCK_ADMINISTRATOR_PASSWORD=${ROCK_ADMINISTRATOR_PASSWORD}
    - ROCK_MANAGER_NAME=${ROCK_MANAGER_NAME}
    - ROCK_MANAGER_PASSWORD=${ROCK_MANAGER_PASSWORD}
    - ROCK_USER_NAME=${ROCK_USER_NAME}
    - ROCK_USER_PASSWORD=${ROCK_USER_PASSWORD}
    - ROCK_ID=${ROCK_ID}
    - ROCK_CLUSTER=${ROCK_CLUSTER}
    - ROCK_TAGS=${ROCK_TAGS}
    # for self-registration
    #- ROCK_SERVER=${ROCK_SERVER}
    #- ROCK_OPAL_SERVER=${ROCK_OPAL_SERVER}
    #- ROCK_OPAL_TOKEN=${ROCK_OPAL_TOKEN}
  volumes:
    - ${PROJECT_HOME}/rock_home:/srv
```

Then environment variables that are exposed by this image are:

Environment Variable	Description
JAVA_OPTS	JVM options.
ROCK_ADMINISTRATOR_NAME	Administrator user name, optional and set at first start.
ROCK_ADMINISTRATOR_PASSWORD	Administrator user password, optional and set at first start.
ROCK_MANAGER_NAME	Manager user name, optional and set at first start.
ROCK_MANAGER_PASSWORD	Manager user password, optional and set at first start.
ROCK_USER_NAME	Regular user name, optional and set at first start.
ROCK_USER_PASSWORD	Regular user password, optional and set at first start.
ROCK_ID	Rock node ID. Make sure it is unique in the cluster.
ROCK_CLUSTER	Cluster of R servers name. Default is “default”.
ROCK_TAGS	Comma separated tag names, optional. Default tag list is empty.
ROCK_SERVER	Self-registration: Rock server public address that will be sent to service registries (including Opal).
ROCK_OPAL_SERVER	Self-registration: Opal server address to register to.
ROCK_OPAL_TOKEN	Self-registration: Opal’s app self-registration token.

Note: When no valid user is defined, the default user setup is applied: user `administrator` with password `password` (with administrator role).

2.3 Upgrade

The upgrade procedures are handled by the application itself.

2.3.1 Debian Package Upgrade

If you installed Rock via the Debian package, you may update it using the command:

```
apt-get install rock
```

2.3.2 RPM Package Upgrade

If you installed Rock via the RPM package, you may update it using the command:

```
yum install rock
```

2.3.3 Zip Distribution Upgrade

Follow the Installation of Rock Zip distribution above but make sure you don't overwrite your rock-home directory.

2.4 Execution

2.4.1 Server launch

Service

When Rock is installed through a Debian/RPM package, Rock server can be managed as a service.

Options for the Java Virtual Machine can be modified if Rock service needs more memory. To do this, modify the value of the environment variable `JAVA_ARGS` in the file **/etc/default/rock**.

Main actions on Rock service are: `start`, `stop`, `status`, `restart`. For more information about available actions on Rock service, type:

```
service rock help
```

The Rock service log files are located in **/var/log/rock** directory.

Manually

The Rock server can be launched from the command line. The environment variable `ROCK_HOME` needs to be setup before launching Rock manually.

Environment variable	Required	Description
`ROCK_HOME`	yes	Path to the Rock "home" directory.
`JAVA_OPTS`	no	Options for the Java Virtual Machine. For example: *-Xmx1024m -XX:MaxPermSize=256m*

To change the defaults update: `bin/rock` or `bin/rock.bat`

Make sure Command Environment is setup and execute the command line (bin directory is in your execution PATH)):

```
rock
```

Executing this command upgrades the Rock server and then launches it.

The Rock server log files are located in **ROCK_HOME/logs** directory. If the logs directory does not exist, it will be created by Rock.

2.4.2 Usage

To access Rock with a web browser the following urls may be used (port numbers may be different depending on HTTP Server Configuration):

- http://localhost:8085 will provide a connection without encryption.

2.4.3 Troubleshooting

If you encounter an issue during the installation and you can't resolve it, please report it in our Rock Issue Tracker.

Rock logs can be found in **/var/log/rock**. If the installation fails, always refer to this log when reporting an error.

CHAPTER

THREE

CONFIGURATION

The file **ROCK_HOME/conf/application.yml** is to be edited to match your server needs. This file is written in YAML format allowing to specify a hierarchy within the configuration keys. The YAML format uses indentations to express the different levels of this hierarchy. The file is already pre-filled with default values (to be modified to match your configuration), just be aware that you should not modify the indentations. In the following documentation, the configuration keys will be presented using the dot-notation (levels are separated by dots) for readability.

After this configuration file has been updated, restart the application so that new settings are effective.

3.1 HTTP Server Configuration

Rock server is a web application and as such, you need to specify on which ports the web server should listen to incoming requests.

Property	Description
`server.port`	HTTP port number. Generally speaking this port should not be exposed to the web. Use proxy instead (Apache2, Nginx, etc.).

Details about the `server.*` properties can be found in the official Spring Boot - Server Properties documentation. More specifically, the `server.ssl.*` properties allows to setup a HTTPS communication channel (default is HTTP). Note also that Rock uses Jetty as the internal web server.

3.2 Cluster Node Configuration

Rock server identification, required when clusters are built.

Property	Description
`node.id`	Rock server unique identifier, unique in the cluster.
`node.cluster`	Name of the cluster to which the R server belongs. Default is “default”.
`node.tags`	Comma separated tag names, for informative purpose.
`node.server`	Public URL of this R server, required for self-registration only.

3.2.1 Consul Configuration

Consul provides a service discovery and health checking infrastructure. It can be used to make Rock servers available to thrid-party applications. These settings allow Rock to self-register itself against a Consul server.

Property	Description
`consul.server`	Consul server URL.
`consul.token`	Consul registration token (highly recommended in production)
`consul.interval`	Rock service interval check in seconds.

3.2.2 Opal Configuration

Opal has an internal App registry to which Rock can self-register itself.

Property	Description
`opal.server`	Opal server URL.
`opal.token`	Opal app registration token (required).

3.3 R Server Configuration

R environment properties.

Property	Description
`r.exec`	File path to the R executable.
`r.repos`	Comma separated list of URLs to R CRAN repositories, to install packages.
`r.sessionTimeout`	Managed R sessions timeout, in minutes. Any R sessions without activity during this laps of time will be removed automatically. Default is `240` (4 hours). If the value is <0 no timeout action is applied.

3.4 Users Configuration

Property	Description
`security.users`	List of users, with attributes `id` (user name), `secret` (user password) and `roles` (comma separated list of role names: `administrator`, `manager` or `user`). See default users for syntax example.

Note: When no valid user is defined, the default user setup is applied: user `administrator` with password `password` (with administrator role).

The user passwords can be hashed, using the Bcrypt algorithm. In this case, the hashed password will be preceded by the prefix **{bcrypt}**. Make sure also to quote the **secret** value so that the $ signs do not get interpreted. There are many bcrypt hashers available online, for instance bcrypt.online. As an example:

```
security:
  users:
    # administrator, can do all
    - id: administrator
```

(continues on next page)

(continued from previous page)

```
    secret: "{bcrypt}$2y$10$Ds/CB6jlY5a4/NU4.RvRI.9oZ16Bp6hx/Xcct1c2XFwYRdMbTHJVu"
    roles: administrator
```

3.5 AppArmor Configuration

Rock can apply an RAppArmor profile on R session creation. Requires RAppArmor to be properly installed and configured.

Property	Description
`security.apparmor.enabled`	Enable/disable the RAppArmor functionality.
`security.apparmor.profile`	The name the AppArmor profile to apply on R session creation.
`security.apparmor.strict`	Whether the AppArmor profile is to be applied to the administrator as well. Default is `true`.

3.6 R Configuration

The file **ROCK_HOME/conf/Rprofile.R** will be executed when the R server main process is started. For an enhanced security (mainly for protecting the server from abusive usage), you can limit the host's resources usage with the unix R package and its rlimit functions.

CHAPTER

FOUR

REST API INTRODUCTION

The Rock REST API has different scopes:

- Service discovery
- R server management
  - Status/start/stop
  - R packages management
- R sessions management and usage
  - R operations
  - File management
  - R commands management

4.1 Authentication

Rock supports only Basic authentication, which consists of a HTTP request's header field in the form of `Authorization: Basic <credentials>`, where credentials is the Base64 encoding of user's (or application's) ID and password joined by a single colon `:`.

Using cURL, it is as simple as providing the –user option:

```
curl --user <id>:<password> [...]
```

4.2 Authorization

Authorizations are role based. The built-in roles are:

Role	Description
`administrator`	Can do anything.
`manager`	Can only manage the R server: R process and R packages. Creating and using R sessions is not permitted.
`user`	Can only create and use its own R sessions. R operations may be limited by the *AppArmor Configuration*.

4.3 Clients

Usage examples provided are based on cURL, a command line client, and rockr a R package. This latter uses the described API slightly differently, as transferred objects are binary serialized R objects instead of JSON ones.

Based on web standards (HTTP, JSON) this API can be used from any programming languages: Opal for instance is a Java application using the Rock services.

CHAPTER

FIVE

SERVICE

These resources are public and are mainly for service discovery operations (identification and availability checks).

5.1 Information

GET /_info

Get the server public information, mainly to allow service discovery.

This entry point does not require any authentication.

Example request

```
curl https://rock-demo.obiba.org/_info
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "name": "rock-demo",
  "type": "rock",
  "tags": [
    "default"
  ]
}
```

Request Headers

- Accept – */*

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – Server is alive and functional.
- 500 Internal Server Error – Server is alive but not functional.

5.2 Check

GET /_check

Get whether the server is functional, to be used for periodic service availability checks. No content is returned, only the response status is meaningful: `200 OK` is positive, whereas any other response indicates a problem.

This entry point does not require any authentication.

Example request

```
curl https://rock-demo.obiba.org/_check
```

Example response

```
HTTP/1.1 200 OK
```

Status Codes

- 200 OK – Server is alive and functional.
- 500 Internal Server Error – Server is alive but not functional.

CHAPTER

SIX

R SERVER

These resources are for managing the R server process running in the background: status, start and stop.

6.1 Status

GET /rserver

Get the server detailed status, including R server and host's system information.

This entry point requires *Authentication* of a user with `administrator` or `manager` role.

Example requests

Using cURL

```
curl --user administrator:password https://rock-demo.obiba.org/rserver
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.status(conn)
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "id": "rock-demo",
  "version": "4.0.3",
  "encoding": "utf8",
  "tags": [
    "default"
  ],
  "running": true,
  "sessions": {
    "total": 0,
    "busy": 0
  },
  "system": {
```

(continues on next page)

(continued from previous page)

```
    "cores": 16,
    "freeMemory": 8938140
  }
}
```

Response JSON Object

- **id** (*string*) – The server identifier.
- **version** (*string*) – The R version.
- **encoding** (*string*) – String encoding when communicating with the R server.
- **tags** (*strings*) – Array of strings, used to identify to what is the R server cluster.
- **running** (*boolean*) – Whether the R server is running or not.
- **sessions.total** (*integer*) – Number of R sessions (active of waiting).
- **sessions.busy** (*integer*) – Number of active R sessions (i.e. an R expression is being executed).
- **system.cores** (*integer*) – Number of host's system cores. If the number of cores cannot be retrieved (when R server is stopped for instance), `-1` is returned.
- **system.freeMemory** (*integer*) – Available host's system memory in KB. If the available memory cannot be retrieved (when R server is stopped for instance), `-1` is returned.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `*/*`

Response Headers

- Content-Type – `application/json`

Status Codes

- 200 OK – Server is alive and functional.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 500 Internal Server Error – Server is alive but not functional.

6.2 Start

`PUT /rserver`

Start the R server. Ignored if the R server is already running.

This entry point requires *Authentication* of a user with `administrator` or `manager` role.

Example requests

Using cURL

```
curl --user administrator:password -X PUT https://rock-demo.obiba.org/rserver
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.start(conn)
```

Request Headers

- Authorization – As described in the *Authentication* section

Status Codes

- 200 OK – Server is alive and functional.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 500 Internal Server Error – Server is alive but not functional.

6.3 Stop

`DELETE /rserver`

Stop the R server. Ignored if the R server is not running. Be aware that all R sessions will be terminated.

This entry point requires *Authentication* of a user with `administrator` or `manager` role.

Example requests

Using cURL

```
curl --user administrator:password -X DELETE https://rock-demo.obiba.org/rserver
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.stop(conn)
```

Request Headers

- Authorization – As described in the *Authentication* section

Status Codes

- 200 OK – Server is alive and functional.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 500 Internal Server Error – Server is alive but not functional.

6.4 Log

GET /rserver/_log?limit=(int: *max_lines*)

Download the last lines of the R server console output. Can be useful when debugging R problems.

This entry point requires *Authentication* of a user with `administrator` or `manager` role.

Example requests

Using cURL

```
curl --user administrator:password -H "Accept: text/plain" https://rock-demo.obiba.
↪org/rserver/_log?limit=100
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.log(conn, 100)
```

Query Parameters

- **limit** (*integer*) – The maximum number of lines to tail from the R server log. Default is `1000`.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `text/plain`

Response Headers

- Content-Type – `text/plain`

Status Codes

- 200 OK – Server is alive and functional.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 500 Internal Server Error – Server is alive but not functional.

CHAPTER

SEVEN

R SERVER PACKAGES

These resources are for managing the R packages installed in the R server.

7.1 List

GET /rserver/packages

List the installed packages. The returned data structure is a matrix.

This entry point requires *Authentication* of a user with `administrator` or `manager` role.

Example requests

Using cURL

```
curl --user administrator:password https://rock-demo.obiba.org/rserver/packages
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.packages(conn)
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "rowNames": [
    "annotate",
    "AnnotationDbi",
    "AnnotationFilter",
    "backports",
    "base64",
    "..."
  ],
  "columnNames": [
    "Package",
    "LibPath",
    "Version",
```

(continues on next page)

(continued from previous page)


```
    "Priority",
    "Depends",
    "Imports",
    "..."
  ],
  "rows": [
    [
      "annotate",
      "/home/yannick/R/library",
      "1.68.0",
      null,
      "R (>= 2.10), AnnotationDbi (>= 1.27.5), XML",
      "Biobase, DBI, xtable, graphics, utils, stats, methods,\nBiocGenerics (>= 0.
↪13.8), httr",
      null,
      "hgu95av2.db, genefilter, Biostrings (>= 2.25.10), IRanges,\nrae230a.db,␣
↪rae230aprobe, tkWidgets, GO.db, org.Hs.eg.db,\norg.Mm.eg.db, hom.Hs.inp.db,␣
↪humanCHRLOC, Rgraphviz, RUnit,",
      null,
      "Artistic-2.0",
      null,
      null,
      null,
      null,
      "no",
      "4.0.3",
      "Annotation for microarrays",
      "Using R enviroments for annotation.",
      "R. Gentleman",
      "Bioconductor Package Maintainer <maintainer@bioconductor.org>",
      "2020-10-27",
      null,
      null,
      null
    ],
    [
      "..."
    ]
  ]
}
```


Response JSON Object

- **rowNames** (*strings*) – Array of R package names.
- **columnNames** (*strings*) – Array of field names from the `DESCRIPTION` files.
- **rows** (*arrays*) – Array of arrays of strings, each value corresponds to the row (=package) name and column (=field) name.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `*/*`

Response Headers

- Content-Type – `application/json`

Status Codes

- 200 OK – Server is alive and functional.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 500 Internal Server Error – Package list could not be retrieved, when R server is not running for instance.

7.2 Update

`PUT /rserver/packages`

Update all CRAN R packages.

This entry point requires *Authentication* of a user with `administrator` or `manager` role.

Example requests

Using cURL

```
curl --user administrator:password -X PUT https://rock-demo.obiba.org/rserver/
↪packages
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.packages_update(conn)
```

Request Headers

- Authorization – As described in the *Authentication* section

Status Codes

- 200 OK – Operation was successful.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 500 Internal Server Error – An error occurred, when R server is not running for instance.

7.3 Remove

DELETE /rserver/packages?name=(string: *package_names*)

Remove specified R packages.

This entry point requires *Authentication* of a user with `administrator` or `manager` role.

Example requests

Using cURL

```
curl --user administrator:password -X DELETE https://rock-demo.obiba.org/rserver/
↪packages?name=annotate,rlang
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.packages_rm(conn, c("annotate", "rlang"))
```

Query Parameters

- **name** (*string*) – One or more R package names to remove, comma separated.

Request Headers

- Authorization – As described in the *Authentication* section

Status Codes

- 204 No Content – Operation was completed. It could have failed silently.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 500 Internal Server Error – An error occurred, when R server is not running for instance.

7.4 Install

POST /rserver/packages?name=(string: *package_name*)**[&manager=string:** *repo_name*`]`**[&ref=string:** *ref_id*`]`

Install a R package from CRAN, GitHub or Bioconductor.

This entry point requires *Authentication* of a user with `administrator` or `manager` role.

Example requests

Using cURL

```
curl --user administrator:password -X POST https://rock-demo.obiba.org/rserver/
↪packages?name=annotate&manager=cran
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.package_install(conn, "annotate", manager = "cran")
```

Query Parameters

- **name** (*string*) – The R package name to install.
- **ref** (*string*) – The Git reference: branch or tag name, commit number. Applies to GitHub repository only. Default is `master`.
- **manager** (*string*) – The R package repository type: `cran`, `github/gh` or `bioconductor/bioc`. Default is `cran`.

Request Headers

- Authorization – As described in the *Authentication* section

Status Codes

- 204 No Content – Operation was successful.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 500 Internal Server Error – An error occurred, when R server is not running for instance.

7.5 DataSHIELD

GET /rserver/packages/_datashield

Get the installed DataSHIELD R packages with their settings. The returned data structure is an object with one entry per DataSHIELD package.

This entry point requires *Authentication* of a user with `administrator` or `manager` role.

Example requests

Using cURL

```
curl --user administrator:password https://rock-demo.obiba.org/rserver/packages/_
↪datashield
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.packages_datashield(conn)
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
```

(continues on next page)

(continued from previous page)


```
  "dsBase": {
    "AggregateMethods": [
      "asFactorDS1",
      "asListDS",
      "boxPlotGGDS",
      "checkNegValueDS",
      "classDS",
      "corTestDS",
      "corDS",
      "covDS",
      "dataFrameSubsetDS1",
      "densityGridDS",
      "..."
    ],
    "AssignMethods": [
      "absDS",
      "asCharacterDS",
      "asDataMatrixDS",
      "asFactorDS",
      "asFactorDS2",
      "asIntegerDS",
      "asListDS",
      "asLogicalDS",
      "..."
    ],
    "Options": [
      "datashield.privacyLevel=5",
      "default.nfilter.glm=0.33",
      "default.nfilter.kNN=3",
      "default.nfilter.string=80",
      "default.nfilter.subset=3",
      "default.nfilter.stringShort=20",
      "default.nfilter.tab=3",
      "default.nfilter.noise=0.25",
      "default.nfilter.levels=0.33"
    ]
  },
  "resourcer": {
    "AssignMethods": [
      "as.resource.data.frame",
      "as.resource.object",
      "as.resource.tbl"
    ]
  }
}
```


Response JSON Object

- **AggregateMethods** (*strings*) – Array of aggregation function names or name mappings.
- **AssignMethods** (*strings*) – Array of assign function names or name mappings.
- **Options** (*strings*) – Array of R options name and value.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `*/*`

Response Headers

- Content-Type – `application/json`

Status Codes

- 200 OK – Server is alive and functional.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 500 Internal Server Error – Package list could not be retrieved, when R server is not running for instance.

CHAPTER

EIGHT

R SERVER PACKAGE

These resources apply to a single R package installed in the R server.

8.1 Description

GET /rserver/package/(string: name)

Get the description of the R package, i.e. all the entries from the DESCRIPTION file.

This entry point requires *Authentication* of a user with `administrator` or `manager` role.

Example requests

Using cURL

```
curl --user administrator:password https://rock-demo.obiba.org/rserver/package/rlang
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.package(conn, "rlang")
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "name": "rlang",
  "Description": "A toolbox for working with base types, core R features\n  like␣
↪the condition system, and core 'Tidyverse' features like tidy\n  evaluation.",
  "Enhances": "winch",
  "Built": "R 4.0.3; x86_64-pc-linux-gnu; 2021-02-22 08:25:16 UTC; unix",
  "License": "MIT + file LICENSE",
  "Imports": "utils",
  "ByteCompile": "true",
  "URL": "https://rlang.r-lib.org, https://github.com/r-lib/rlang",
  "RoxygenNote": "7.1.1",
  "BugReports": "https://github.com/r-lib/rlang/issues",
  "LazyData": "true",
```

(continues on next page)

(continued from previous page)


```
  "Maintainer": "Lionel Henry <lionel@rstudio.com>",
  "Config/testthat/edition": "3",
  "Version": "0.4.10",
  "Suggests": "cli, covr, crayon, glue, magrittr, methods, pak, pillar,\nrmarkdown,␣
↪testthat (>= 3.0.0), vctrs (>= 0.2.3), withr",
  "NeedsCompilation": "yes",
  "Authors@R": "c(\n    person(\"Lionel\", \"Henry\", ,\"lionel@rstudio.com\", c(\
↪"aut\", \"cre\")),\n    person(\"Hadley\", \"Wickham\", ,\"hadley@rstudio.com\", \
↪"aut\"),\n    person(given = \"mikefc\",\n           email = \
↪"mikefc@coolbutuseless.com\", \n           role = \"cph\", \n           comment =␣
↪\"Hash implementation based on Mike's xxhashlite\"),\n    person(given = \"Yann\",
↪\n           family = \"Collet\",\n           role = \"cph\", \n           ␣
↪comment = \"Author of the embedded xxHash library\"),\n    person(\"RStudio\",␣
↪role = \"cph\")\n    )",
  "Date/Publication": "2020-12-30 15:00:02 UTC",
  "Packaged": "2020-12-18 09:35:30 UTC; lionel",
  "Biarch": "true",
  "Title": "Functions for Base Types and Core R and 'Tidyverse' Features",
  "Encoding": "UTF-8",
  "Repository": "CRAN",
  "Author": "Lionel Henry [aut, cre],\n  Hadley Wickham [aut],\n  mikefc [cph]␣
↪(Hash implementation based on Mike's xxhashlite),\n  Yann Collet [cph] (Author of␣
↪the embedded xxHash library),\n  RStudio [cph]",
  "Package": "rlang",
  "Depends": "R (>= 3.3.0)"
}
```


Response JSON Object

- **name** (*string*) – R package name.
- **<field>** (*string*) – Field value from the `DESCRIPTION` file.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `*/*`

Response Headers

- Content-Type – `application/json`

Status Codes

- 200 OK – Server is alive and functional.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 404 Not Found – Package with provided name could not be found.
- 500 Internal Server Error – Package description could not be retrieved, when R server is not running for instance.

8.2 Remove

DELETE /rserver/package/(string: *name*)

Remove the R package.

This entry point requires *Authentication* of a user with `administrator` or `manager` role.

Example requests

Using cURL

```
curl --user administrator:password -X DELETE https://rock-demo.obiba.org/rserver/
↪package/rlang
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.package_rm(conn, "rlang")
```

Request Headers

- Authorization – As described in the *Authentication* section

Status Codes

- 204 No Content – Operation was completed. It could have failed silently.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 500 Internal Server Error – An error occurred, when R server is not running for instance.

CHAPTER

NINE

R SESSIONS

These resources manage the stateful R sessions living in the R server. Some operations may be limited depending on the user role.

9.1 List

GET /r/sessions[?subject=(string: *user_name*)]

List the R sessions.

This entry point requires *Authentication* of a user. Users with `administrator` or `manager` role will be able to list other users sessions. Regular users can only list own R sessions.

Example requests

Using cURL

```
curl --user administrator:password https://rock-demo.obiba.org/r/sessions
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.sessions(conn)
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "id": "810cfda6-d0f5-472e-8796-0ce6905499d8",
    "subject": "user",
    "busy": false,
    "createdDate": "2021-02-24 09:11:08",
    "lastAccessDate": "2021-02-24 09:11:15"
  }
]
```

Query Parameters

- **subject** (*string*) – To filter sessions by their subject (owner of the session). Ignored when user does not have `administrator` or `manager` role.

Response JSON Object

- **id** (*string*) – R session unique ID.
- **subject** (*string*) – User name owning the R session.
- **busy** (*boolean*) – Whether an R operation is being executed.
- **createdDate** (*date*) – Date of creation.
- **lastAccessDate** (*date*) – Last time the R session was accessed, used to garbage collect sessions after some timeout.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `*/*`

Response Headers

- Content-Type – `application/json`

Status Codes

- 200 OK – Server is alive and functional.
- 401 Unauthorized – User is not authenticated.
- 500 Internal Server Error – Sessions list could not be retrieved.

9.2 Remove

`DELETE /r/sessions`

Terminate all the R sessions.

This entry point requires *Authentication* of a user with `administrator` or `manager` role.

Example requests

Using cURL

```
curl --user administrator:password -X DELETE https://rock-demo.obiba.org/r/sessions
```

Request Headers

- Authorization – As described in the *Authentication* section

Status Codes

- 204 No Content – Operation was completed.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 500 Internal Server Error – An error occurred.

9.3 Create

POST /r/sessions

Create a R session, which will be associated to the requesting user.

This entry point requires *Authentication* of a user with `administrator` or `user` role.

Example requests

Using cURL

```
curl --user user:password -X POST https://rock-demo.obiba.org/r/sessions
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="user", password="password", url = "https://rock-
↪demo.obiba.org")
rockr.open(conn)
```

Example response

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://rock-demo.obiba.org/r/session/810cfda6-d0f5-472e-8796-0ce6905499d8

{
  "id": "810cfda6-d0f5-472e-8796-0ce6905499d8",
  "subject": "user",
  "busy": false,
  "createdDate": "2021-02-24 09:11:08",
  "lastAccessDate": "2021-02-24 09:11:15"
}
```

Response JSON Object

- **id** (*string*) – R session unique ID.
- **subject** (*string*) – User name owning the R session.
- **busy** (*boolean*) – Whether an R operation is being executed.
- **createdDate** (*date*) – Date of creation.
- **lastAccessDate** (*date*) – Last time the R session was accessed, used to garbage collect sessions after some timeout.

Request Headers

- Authorization – As described in the *Authentication* section

Status Codes

- 204 No Content – Operation was completed.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role for this operation.
- 500 Internal Server Error – An error occurred.

CHAPTER

TEN

R SESSION

These resources apply to a single R session living in the R server. Only R session owner or a user with `administrator` role can operate.

10.1 Status

GET /r/session/(string: *id*)

Get the R session status.

This entry point requires *Authentication* of a user. Users with `administrator` or `manager` role will be able to get other users session. Regular users can only get own R session.

Example requests

Using cURL

```
curl --user user:password https://rock-demo.obiba.org/r/session/810cfda6-d0f5-472e-
↪8796-0ce6905499d8
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "id": "810cfda6-d0f5-472e-8796-0ce6905499d8",
  "subject": "user",
  "busy": false,
  "createdDate": "2021-02-24 09:11:08",
  "lastAccessDate": "2021-02-24 09:11:15"
}
```

Response JSON Object

- **id** (*string*) – R session unique ID.
- **subject** (*string*) – User name owning the R session.
- **busy** (*boolean*) – Whether an R operation is being executed.
- **createdDate** (*date*) – Date of creation.
- **lastAccessDate** (*date*) – Last time the R session was accessed, used to garbage collect sessions after some timeout.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `*/*`

Response Headers

- Content-Type – `application/json`

Status Codes

- 200 OK – Server is alive and functional.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role or permission for this operation.
- 404 Not Found – Session could not be found.
- 500 Internal Server Error – Session could not be accessed.

10.2 Remove

`DELETE /r/session/(string: id)`

Terminate the R session, free memory and file system usage.

This entry point requires *Authentication* of a user. Users with `administrator` or `manager` role will be able to remove other users session. Regular users can only remove own R session.

Example requests

Using cURL

```
curl --user administrator:password -X DELETE https://rock-demo.obiba.org/r/session/
↪810cfda6-d0f5-472e-8796-0ce6905499d8
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="administrator", password="password", url = "https://
↪rock-demo.obiba.org")
rockr.open(conn)
rockr.close(conn)
```

Request Headers

- Authorization – As described in the *Authentication* section

Status Codes

- 204 No Content – Operation was completed. It could have failed silently.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role or permission for this operation.
- 404 Not Found – Session could not be found.
- 500 Internal Server Error – An error occurred.

10.3 Assign

POST /r/session/(string: *id*)**/_assign?s=**
string: *symbol*[**&async=boolean:** *async*]

Assign an R expression to *symbol*. The R expression is the body of the request. The R expression can be the string representation of the data or a function call.

This entry point requires *Authentication* of a user. Users with `administrator` role will be able to use other users session. Regular users can only use own R session.

Example requests

Using cURL

```
curl --user user:password -H "Content-Type: application/x-rscript" --data "getwd()"␣
↪https://rock-demo.obiba.org/r/session/810cfda6-d0f5-472e-8796-0ce6905499d8/_
↪assign?s=x
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="user", password="password", url = "https://rock-
↪demo.obiba.org")
rockr.open(conn)
rockr.assign(conn, "x", quote(getwd()))
rockr.assign(conn, "n", 123)
rockr.assign(conn, "str", "abc")
```

Example response

When *async* parameter is `true`, a *Command* is created and put in the execution queue.

```
HTTP/1.1 201 Created
Location: https://rock-demo.obiba.org/r/session/810cfda6-d0f5-472e-8796-
↪0ce6905499d8/command/605fc0a4-4e41-40eb-bf40-89c2d3bb17fa-3

{
  "id": "605fc0a4-4e41-40eb-bf40-89c2d3bb17fa-3",
  "status": "IN_PROGRESS",
  "finished": false,
  "createdDate": "2021-02-24T17:38:14.929+00:00",
  "startDate": "2021-02-24T17:38:14.929+00:00",
  "endDate": "2021-02-24T17:38:14.930+00:00",
  "withError": false,
  "withResult": false,
  "script": "base::assign('x', getwd())"
}
```

Query Parameters

- **s** (*string*) – The R symbol name to assign.
- **async** (*boolean*) – Whether the R operation is to be put in a command queue for latter execution, in which case a *Command* object will be returned (see *Commands*). Default is `false`.

Request Headers

- Authorization – As described in the *Authentication* section
- Content-Type – `application/x-rscript`

Status Codes

- 200 OK – Operation was completed.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role or permission for this operation.
- 404 Not Found – Session could not be found.
- 500 Internal Server Error – An error occurred.

10.4 Evaluate

POST /r/session/(string: *id*)/_eval[?async=boolean: *async*]

Evaluate an R expression. The R expression is the body of the request. The R expression can be the string representation of the data or a function call. The returned value can be a primitive type or JSON array/object. In the latter case, make sure that the R expression call returns a value that can be serialized using jsonlite::toJSON(). If *toJSON()* fails, instead of raising an error, Rock will fallback to jsonlite::serializeJSON() which is more robust (and also quite verbose).

This entry point requires *Authentication* of a user. Users with `administrator` role will be able to use other users session. Regular users can only use own R session.

Example requests

Using cURL

```
curl --user user:password -H "Content-Type: application/x-rscript" --data "getwd()"␣
↪https://rock-demo.obiba.org/r/session/810cfda6-d0f5-472e-8796-0ce6905499d8/_eval
# note: R.version value cannot be stringified in JSON as-is
curl --user user:password -H "Content-Type: application/x-rscript" --data "as.
↪list(unlist(R.version))" https://rock-demo.obiba.org/r/session/810cfda6-d0f5-472e-
↪8796-0ce6905499d8/_eval
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="user", password="password", url = "https://rock-
↪demo.obiba.org")
rockr.open(conn)
rockr.eval(conn, quote(R.version))
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "platform": "x86_64-pc-linux-gnu",
  "arch": "x86_64",
  "os": "linux-gnu",
```

(continues on next page)

(continued from previous page)


```
  "system": "x86_64, linux-gnu",
  "status": "",
  "major": "4",
  "minor": "0.4",
  "year": "2021",
  "month": "02",
  "day": "15",
  "svn rev": "80002",
  "language": "R",
  "version.string": "R version 4.0.4 (2021-02-15)",
  "nickname": "Lost Library Book"
}
```


Query Parameters

- **async** (*boolean*) – Whether the R operation is to be put in a command queue for latter execution, in which case a *Command* object will be returned (see *Commands*) in place of the evaluation result. Default is `false`.

Request Headers

- Authorization – As described in the *Authentication* section
- Content-Type – `application/x-rscript`
- Accept – `*/*`

Response Headers

- Content-Type – `application/json`

Status Codes

- 200 OK – Operation was completed.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role or permission for this operation.
- 404 Not Found – Session could not be found.
- 500 Internal Server Error – An error occurred.

10.5 Files

These resources are for exchanging files between the client and an R session's workspace.

10.5.1 Upload

POST /r/session/(string: *id*)**/_upload?[path=**
string: *path*]**[&overwrite=boolean:** *overwrite*]**[&temp=boolean:** *temp*]

Upload a file at *path*. If *path* is not specified, the uploaded file name will be used. Note that the *path* root folder is ever the R session original working directory or its temporary directory (if *temp* is `true`). This means that any attempt to upload a file outside of the R session file scope will fail.

This entry point requires *Authentication* of a user. Users with `administrator` role will be able to use other users session. Regular users can only use own R session.

Example requests

Using cURL

```
curl --user user:password -F "file=@some/local/file.ext" https://rock-demo.obiba.
↪org/r/session/810cfda6-d0f5-472e-8796-0ce6905499d8/_upload?overwrite=true
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="user", password="password", url = "https://rock-
↪demo.obiba.org")
rockr.open(conn)
rockr.file_upload(conn, source = "some/local/file.ext", overwrite = TRUE)
```

Query Parameters

- **path** (*string*) – The destination path relative to the root directory (defined by the *temp* parameter). Any subfolders will be created automatically. If this parameter is missing, the uploaded file name will be used.
- **overwrite** (*boolean*) – Whether to overwrite the destination file if it already exists. Default is `false`.
- **temp** (*boolean*) – Whether the root directory is the temporary folder of the R session, otherwise it will be the original working directory. Default is `false`.

Request Headers

- Authorization – As described in the *Authentication* section
- Content-Type – `multipart/form-data`

Status Codes

- 200 OK – Operation was completed.
- 400 Bad Request – If destination file is a folder, exists and cannot be overridden or more generally if it is not valid (attempt to write a file outside of the R session root directory).
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role or permission for this operation.
- 404 Not Found – Session could not be found.
- 500 Internal Server Error – An error occurred.

10.5.2 Download

GET /r/session/(string: *id*)**/_download?path=**
string: *path*[**&temp=boolean:** *temp*]

Download a file located at *path* in the R session root directory. This root directory is ever the R session original working directory or its temporary directory (if *temp* is `true`). This means that any attempt to download a file from outside of the R session file scope will fail.

This entry point requires *Authentication* of a user. Users with `administrator` role will be able to use other users session. Regular users can only use own R session.

Example requests

Using cURL

```
curl --user user:password https://rock-demo.obiba.org/r/session/810cfda6-d0f5-472e-
↪8796-0ce6905499d8/_upload?path=some%2Fremote%2Ffile.ext -o file.ext
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="user", password="password", url = "https://rock-
↪demo.obiba.org")
rockr.open(conn)
rockr.file_download(conn, source = "some/remote/file.ext", destination = "file.ext")
```

Query Parameters

- **path** (*string*) – The source path relative to the root directory (defined by the *temp* parameter).
- **temp** (*boolean*) – Whether the root directory is the temporary folder of the R session, otherwise it will be the original working directory. Default is `false`.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `*/*`

Status Codes

- 200 OK – Operation was completed.
- 400 Bad Request – If file does not exists, is a folder or more generally if it is not valid (attempt to access a file outside of the R session root directory).
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role or permission for this operation.
- 404 Not Found – Session could not be found.
- 500 Internal Server Error – An error occurred.

10.6 Commands

These resources are for managing the R operations that are executed asynchronously in an R session. See *Assign* and *Evaluate* requests that propose an *async* query parameter to put the R operation in the execution queue.

10.6.1 List

GET /r/session/(string: *id*)/commands

List the R commands that are either in the processing queue or in the result list.

This entry point requires *Authentication* of a user. Users with `administrator` role will be able to use other users session. Regular users can only use own R session.

Example requests

Using cURL

```
curl --user user:password https://rock-demo.obiba.org/r/session/810cfda6-d0f5-472e-
↪8796-0ce6905499d8/commands
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="user", password="password", url = "https://rock-
↪demo.obiba.org")
rockr.open(conn)
rockr.eval(conn, quote(R.version), async = TRUE)
rockr.assign(conn, "x", quote(getwd()), async = TRUE)
rockr.commands(conn)
```

Example response

```
HTTP/1.1 200 OK

[
  {
    "id": "1",
    "sessionId": "810cfda6-d0f5-472e-8796-0ce6905499d8",
    "status": "COMPLETED",
    "finished": true,
    "createdDate": "2021-02-24T17:55:59.133+00:00",
    "startDate": "2021-02-24T17:55:59.133+00:00",
    "endDate": "2021-02-24T17:55:59.164+00:00",
    "withError": false,
    "withResult": true,
    "script": "R.version"
  },
  {
    "id": "2",
    "sessionId": "810cfda6-d0f5-472e-8796-0ce6905499d8",
    "status": "COMPLETED",
    "finished": true,
    "createdDate": "2021-02-24T17:57:03.129+00:00",
```

(continues on next page)

(continued from previous page)

```
    "startDate": "2021-02-24T17:57:03.130+00:00",
    "endDate": "2021-02-24T17:57:03.131+00:00",
    "withError": false,
    "withResult": false,
    "script": "base::assign('x', getwd())"
  }
]
```

Response JSON Array of Objects

- **id** (*string*) – Command unique ID.
- **status** (*string*) – The status is one of: `PENDING` (command is in the execution queue), `IN_PROGRESS` (command execution is in progress), `COMPLETED` (completion with success) or `FAILED` (completion with failure).
- **finished** (*boolean*) – Whether the command is completed (successfully or not).
- **createdDate** (*string*) – Date of command submission.
- **startDate** (*string*) – Date of the command execution start.
- **endDate** (*string*) – Date of the command execution completion.
- **withError** (*boolean*) – Whether failed completion has an error message.
- **error** (*string*) – Error message.
- **withResult** (*boolean*) – Whether completed command has a result. See *Result*.
- **script** (*string*) – R script associated to the command.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `*/*`

Response Headers

- Content-Type – `application/json`

Status Codes

- 200 OK – Operation was completed.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role or permission for this operation.
- 404 Not Found – Session could not be found.
- 500 Internal Server Error – An error occurred.

10.6.2 Status

GET /r/session/(string: *id*)**/command/**
string: *cmd_id*

Get the status of a command in its R session.

This entry point requires *Authentication* of a user. Users with `administrator` role will be able to use other users session. Regular users can only use own R session.

Example requests

Using cURL

```
curl --user user:password https://rock-demo.obiba.org/r/session/810cfda6-d0f5-472e-
↪8796-0ce6905499d8/command/1
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="user", password="password", url = "https://rock-
↪demo.obiba.org")
rockr.open(conn)
cmd <- rockr.eval(conn, quote(R.version), async = TRUE)
rockr.command(conn, cmd$id)
```

Example response

```
HTTP/1.1 200 OK

{
  "id": "1",
  "sessionId": "810cfda6-d0f5-472e-8796-0ce6905499d8",
  "status": "COMPLETED",
  "finished": true,
  "createdDate": "2021-02-24T17:55:59.133+00:00",
  "startDate": "2021-02-24T17:55:59.133+00:00",
  "endDate": "2021-02-24T17:55:59.164+00:00",
  "withError": false,
  "withResult": true,
  "script": "R.version"
}
```

Response JSON Object

- **id** (*string*) – Command unique ID.
- **status** (*string*) – The status is one of: `PENDING` (command is in the execution queue), `IN_PROGRESS` (command execution is in progress), `COMPLETED` (completion with success) or `FAILED` (completion with failure).
- **finished** (*boolean*) – Whether the command is completed (successfully or not).
- **createdDate** (*string*) – Date of command submission.
- **startDate** (*string*) – Date of the command execution start.
- **endDate** (*string*) – Date of the command execution completion.
- **withError** (*boolean*) – Whether failed completion has an error message.

- **error** (*string*) – Error message.
- **withResult** (*boolean*) – Whether completed command has a result. See *Result*.
- **script** (*string*) – R script associated to the command.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – */*

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – Operation was completed.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role or permission for this operation.
- 404 Not Found – Session or command could not be found.
- 500 Internal Server Error – An error occurred.

10.6.3 Remove

DELETE /r/session/(string: *id*)/command/
string: *cmd_id*

Remove a command, before or after it has been executed. Note that it will not interrupt a running execution.

This entry point requires *Authentication* of a user. Users with administrator role will be able to use other users session. Regular users can only use own R session.

Example requests

Using cURL

```
curl --user user:password -X DELETE https://rock-demo.obiba.org/r/session/810cfda6-
↪d0f5-472e-8796-0ce6905499d8/command/1
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="user", password="password", url = "https://rock-
↪demo.obiba.org")
rockr.open(conn)
cmd <- rockr.eval(conn, quote(R.version), async = TRUE)
rockr.command_rm(conn, cmd$id)
```

Request Headers

- Authorization – As described in the *Authentication* section

Status Codes

- 204 No Content – Operation was completed.
- 401 Unauthorized – User is not authenticated.

- 403 Forbidden – User does not have the appropriate role or permission for this operation.
- 404 Not Found – Session or command could not be found.
- 500 Internal Server Error – An error occurred.

10.6.4 Result

GET /r/session/(string: *id*)**/command/**
string: *cmd_id*/**result[?rm=boolean:** *remove*]**[&wait=boolean:** *wait*]

Extract the result from a completed, successful, command having some result data.

This entry point requires *Authentication* of a user. Users with `administrator` role will be able to use other users session. Regular users can only use own R session.

Example requests

Using cURL

```
curl --user user:password https://rock-demo.obiba.org/r/session/810cfda6-d0f5-472e-
↪8796-0ce6905499d8/command/1/result?wait=true
```

Using R (rockr)

```
library(rockr)
conn <- rockr.connect(username="user", password="password", url = "https://rock-
↪demo.obiba.org")
rockr.open(conn)
cmd <- rockr.eval(conn, quote(R.version), async = TRUE)
rockr.command_result(conn, cmd$id, wait = TRUE)
```

Query Parameters

- **rm** (*boolean*) – Remove the command from the result list after the result download. Default is `true`.
- **wait** (*boolean*) – Whether the command completion should be waited in a blocking way. Default is `false`.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `*/*`

Response Headers

- Content-Type – `application/json`

Status Codes

- 200 OK – Operation was completed.
- 204 No Content – Empty response when command is not completed yet and *wait* parameter is `false`.
- 401 Unauthorized – User is not authenticated.
- 403 Forbidden – User does not have the appropriate role or permission for this operation.
- 404 Not Found – Session or command could not be found.

- 500 Internal Server Error – An error occurred.

CHAPTER

ELEVEN

PARTNERS AND FUNDERS

The development of this application was made possible thanks to the support of our partners and funders:

CHAPTER

TWELVE

SUPPORT

Please visit OBiBa support page.

HTTP ROUTING TABLE

/_check

GET /_check, 22

/_info

GET /_info, 21

/r

GET /r/session/(string:id), 43
GET /r/session/(string:id)/_download?path=(string:path)[&temp=(boolean:temp)], 49
GET /r/session/(string:id)/command/(string:cmd_id), 52
GET /r/session/(string:id)/command/(string:cmd_id)/result[?rm=(boolean:remove)][&wait=(boolean:wait)], 54
GET /r/session/(string:id)/commands, 50
GET /r/sessions[?subject=(string:user_name)], 39
POST /r/session/(string:id)/_assign?s=(string:symbol)[&async=(boolean:async)], 45
POST /r/session/(string:id)/_eval[?async=(boolean:async)], 46
POST /r/session/(string:id)/_upload?[path=(string:path)][&overwrite=(boolean:overwrite)][&temp=(boolean 48
POST /r/sessions, 41
DELETE /r/session/(string:id), 44
DELETE /r/session/(string:id)/command/(string:cmd_id), 53
DELETE /r/sessions, 40

/rserver

GET /rserver, 23
GET /rserver/_log?limit=(int:max_lines), 26
GET /rserver/package/(string: name), 35
GET /rserver/packages, 27
GET /rserver/packages/_datashield, 31
POST /rserver/packages?name=(string:package_name)[&manager=(string:repo_name)][&ref=(string:ref_id)], 30
PUT /rserver, 24
PUT /rserver/packages, 29
DELETE /rserver, 25
DELETE /rserver/package/(string:name), 37
DELETE /rserver/packages?name=(string:package_names), 30