

---

# **libreant Documentation**

***Release 0.5***

**insomnialab**

**Dec 12, 2017**



---

## Contents

---

<b>1</b>	<b>About libreant</b>	<b>3</b>
<b>2</b>	<b>Libreant architecture</b>	<b>5</b>
2.1	How to set up an aggregator . . . . .	5
<b>3</b>	<b>Librarian</b>	<b>7</b>
3.1	Presets system . . . . .	7
<b>4</b>	<b>Sysadmin</b>	<b>11</b>
4.1	Installation . . . . .	11
4.2	Execution . . . . .	12
4.3	Upgrading . . . . .	13
<b>5</b>	<b>How to write documentation</b>	<b>15</b>
5.1	Markup language . . . . .	15
5.2	Documentation directory . . . . .	15
5.3	Documenting code . . . . .	16
<b>6</b>	<b>How to develop</b>	<b>17</b>
6.1	Ingredients . . . . .	17
6.2	Installation . . . . .	17
6.3	Code design . . . . .	18
6.4	Testing . . . . .	19
6.5	Contributing . . . . .	20
<b>7</b>	<b>API</b>	<b>21</b>
7.1	archivant package . . . . .	21
7.2	conf package . . . . .	21
7.3	libreantdb package . . . . .	22
7.4	presets package . . . . .	22
7.5	users package . . . . .	23
<b>8</b>	<b>Libreant changelog</b>	<b>25</b>
8.1	next . . . . .	25
8.2	0.5 . . . . .	25
8.3	0.4 . . . . .	26
8.4	0.3 . . . . .	27

<b>9 Indices and tables</b>	<b>29</b>
<b>Python Module Index</b>	<b>31</b>

Contents:



# CHAPTER 1

---

## About libreant

---

Libreant is a book manager for both digital and paper documents. It can store any kind of digital data actually, not only books. It's db structure makes Libreant highly customizable, documents can be archived by their types with different metadata set, moreover you can create your own preset and choose default descriptors for that kind of volume. The search function looks through the db, and rank matches powered by ElasticSearch. The language of metadata (as title, or description) is a compulsory field, since the db will use it to optimize the search.

Elements into Libreant are defined as volumes, for each volume you can attach many files, usually these files are pdf or book scans. Libreant is built and intended as a federation of nodes, every node is an archive. From a node you can search into friend-nodes, with OpenSearch protocol. Possible extensions into Web are suspended.

Libreant aims to share, find and save books. It can be used by a librarian who needs an archive system or to collect digital items in a file sharing project.

Libreant is created by InsomniLab, a hacklab in Rome. For any doubts, suggestion or similar write to: [insomniab@hacari.org](mailto:insomniab@hacari.org)

Libreant is Ubercool





---

### Libreant architecture

---

Libreant is meant to be a distributed system. Actually, you can even think of nodes as standalone-systems. A node is not aware of other nodes. It is a single point of distribution with no knowledge of other points.

The system that binds the nodes together is the aggregator; an aggregator acts only as a client with respect to the nodes. Therefore multiple aggregators can coexist. This also implies that the node administration does not involve the management of the aggregation mechanism and of the aggregators themselves. Similarly, it is possible to run an aggregator without running a libreant node. As a consequence, a node cannot choose whether to be aggregated or not.

The aggregation mechanism is based on [Opensearch](#), and relies on two mandatory fields:

- the Opensearch [description](#)
- the Opensearch [response](#)

meaning that this entries are mandatory on a node in order to be aggregated. The result component heavily relies on the [relevance](#) extension of the response spec.

We blindly trust this relevance field, so a malicious node could bias the overall result, simply increasing the relevance fields of its entries. In this way, the management of the aggregators implies also the task of checking the fairness of the aggregated nodes.

## 2.1 How to set up an aggregator

1. Install Libreant. Follow the instructions on [Installation](#).
2. Launch Libreant setting the `AGHERANT_DESCRIPTIONS` configuration parameters. Its value should be a list of URLs. Each URL represents the Opensearch description. For Libreant it's located in `/description.xml`, so a typical URL looks like:

`http://your.doma.in/description.xml`

and a typical invocation looks like:

```
libreant --agherant-descriptions "http://your.doma.in/description.xml http://  
↳other.node/description.xml"
```

If you want to aggregate the same libreant instance that you are running, there's a shortcut: just use `SELF`. Here's an example:

```
libreant --agherant-descriptions "SELF http://other.node/description.xml"
```

---

**Note:** Through *agherant* command line program, it's possible to run an aggregator without launching the whole libreant software

---

This chapter is dedicated to librarians, people who manage the libreant node, decide how to structure the database, organize informations and supervise the catalogue.

### 3.1 Presets system

One of the things that make libreant powerful is that there are almost no assumptions and restrictions about informations you can catalog with it. You can use libreant to store digital book, organize physical book metadata, CDs, comics, organization reports, posters and so on.

Stored object informations are organized in a collection of key-values pairs:

```
title:   Heart of Darkness
author:  Joseph Conrad
year:    1899
country: United Kingdom
```

Normally, when users insert new objects in the database they can choose the number and the type of key-values pairs to save, without any restrictions. Language field is the only one information that is always required.

All this freedom could be difficult to administrate, so libreant provide the preset system as a useful tool to help librarians.

#### 3.1.1 Preset

A preset is a set of rules and properties that denote a class of object. For example, if you want to store physical book metadata in your libreant node and for every book you want to remember the date in which you bought that book, in this case you can create a preset for class `bought-book` that has always a property with id `date`.

### 3.1.2 Quick steps creation

To create a new preset you need to create a new json file, populate it and configure libreant to use it.

Every preset is described by one json formatted text file. So in order to create a new preset you need to create a new text file with .json extension. This is the simplest preset you can do:

```
{
  "id": "bought-book",
  "properties": []
}
```

Once you have created all your presets you can use the PRESET\_PATHS configuration variable to make libreant use them. PRESET\_PATHS accepts a list of paths ( strings ), you can pass paths to file or folders containing presets.

Start libreant and go to the add page, you should have a list menu from which you can choose one of your presets. If some of your presets are not listed, you can take a look at log messages to investigate the problem.

### 3.1.3 Preset structure

The preset file has some general fields that describe the matadata of the preset (id, description, etc... ) and a list of properties describing informations that objects belonging to this preset must/should have.

Preset example:

```
{
  "id": "bought-book",
  "allow_upload": false,
  "description": "bought physical book",
  "properties": [{ "id": "title",
    "description": "title of the book",
    "required": true
  },
    { "id": "author",
    "description": "author of the book",
    "required": true
  },
    { "id": "date",
    "description": "date in which book was bought",
    "required": true
  },
    { "id": "genre",
    "description": "genre of the book",
    "required": true,
    "type": "enum",
    "values": ["novel", "scientific", "essay", "poetry"]
  }
]
```

General fields:

Key	Type	Required	Default	Description
id	string	True		id of the preset
description	string	False	""	a brief description of the preset
allow_upload	boolean	False	True	permits upload of files during submission
properties	list	True		list of properties

Property fields:

Key	Type	Required	Default	Description
id	string	True		id of the property
description	string	False	“”	a brief description of the property
required	boolean	False	False	permits to leave this property empty during submission
type	string	False	“string”	the type of this property
values	list	<i>Enum type</i>		used if type is “enum”

### String type

String type properties will appear in the add page as a plain text field.

### Enum type

Enum type properties will appear in the add page as a list of values. Possible values must be placed in *values* field as list of strings. *values* field are required if the type of the same property is “enum”.



### 4.1 Installation

Libreant is written in Python and uses Elasticsearch as the underlying search engine. In the following sections there are the step-by-step guides to install Libreant on different linux-based operating system:

- *Debian, Ubuntu and all the debian based distribution*
- *Arch Linux*

#### 4.1.1 Debian & Ubuntu

##### System dependencies

##### Install Elasticsearch

The recommended way of installing Elasticsearch on debian-based distro is through the official APT repository.

**Note:** If you have any problem installing elasticsearch try to follow the [official deb installation guide](#)

In order to follow the Elasticsearch installation steps we need to install some common packages:

```
sudo apt-get update && sudo apt-get install apt-transport-https wget gnupg ca-  
↪certificates
```

Download and install the Public Signing Key for elasticsearch repo:

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
```

Add elasticsearch repository:

```
echo "deb https://artifacts.elastic.co/packages/6.x/apt stable main" | sudo tee -a /  
↳etc/apt/sources.list.d/elastic-6.x.list
```

And finally you can install the Elasticsearch package with:

```
sudo apt-get update && sudo apt-get install openjdk-8-jre-headless procps_  
↳elasticsearch
```

---

**Note:** The procps provides the `ps` command that is required by the elasticsearch startup script

---

## Install Python

Libreant is going to be installed into a python virtual environment, thus we need to install it:

```
sudo apt-get update && sudo apt-get install python2.7 virtualenv python-wheel
```

## Install libreant

Create a virtual env:

```
virtualenv -p /usr/bin/python2 ve
```

Install libreant and all python dependencies:

```
./ve/bin/pip install libreant
```

## 4.1.2 Arch

Install all necessary packages:

```
sudo pacman -Sy python2 python2-setuptools python2-virtualenv grep procps_  
↳elasticsearch
```

---

**Note:** The procps and grep packages are required by the elasticsearch startup script

---

Create a virtual env:

```
virtualenv2 -p /usr/bin/python2 ve
```

Install libreant and all python dependencies:

```
./ve/bin/pip install libreant
```

## 4.2 Execution

Start elasticsearch service:



```
sudo service elasticsearch start
```

**Note:** If you want to automatically start elasticsearch during bootup:

```
sudo systemctl enable elasticsearch
```

To execute libreant:

```
./ve/bin/libreant
```

## 4.3 Upgrading

Generally speaking, to upgrade libreant you just need to:

```
./ve/bin/pip install -U libreant
```

And restart your instance (see the [Execution](#) section).

Some versions, however, could need additional actions. We will list them all in this section.

### 4.3.1 Upgrade to version 0.5

libreant now supports elasticsearch 2. If you were already using libreant 0.4, you were using elasticsearch 1.x. You *can* continue using it if you want. The standard upgrade procedure is enough to have everything working. However, we suggest you to upgrade to elasticsearch2 sooner or later.

#### Step 1: stop libreant

For more info, see [Execution](#); something like `pkill libreant` should do

#### Step 2: upgrade elasticsearch

Just apply the steps in [Installation](#) section as if it was a brand new installation.

---

**Note:** If you are using archlinux, you’ve probably made pacman ignore elasticsearch package updates. In order to install the new elasticsearch version you must remove the `IgnorePkg elasticsearch` line in `/etc/pacman.conf` *before* trying to upgrade.

---

#### Step 3: upgrade DB contents

Libreant ships a tool that will take care of the upgrade. You can run it with `./ve/bin/libreant-db upgrade`.

This tool will give you information on the current DB status and ask you for confirmation before proceeding to real changes. Which means that you can run it without worries, you’re still in time for answering “no” if you change your mind.

The upgrade tool will ask you about converting entries to the new format, and upgrading the index mapping (in elasticsearch jargon, this is somewhat similar to what a `TABLE SCHEMA` is in `SQL`)

---

## How to write documentation

---

We care a lot about documentation. So this chapter is both about technical reference and guidelines.

### 5.1 Markup language

Documentation is written using `restructuredText`; it's a very rich markup language, so learning it all may be difficult. You can start reading a [quick guide](#); you can then pass to a slightly [longest guide](#).

As with all the code, you can learn much just reading pre-existing one. So go to next section and you'll know where it is placed.

### 5.2 Documentation directory

Documentation is placed in `doc/source/` in libreant repository. Yes, it's just a bunch of `.rst` files. The main one is `index.rst`, and his main part is the `toctree` directive; the list below it specifies the order in which to include all the other pages.

---

**Note:** If you are trying to add a new page to the documentation, remember to add its filename to the `toctree` in `index.rst`

---

To build html documentation from it, you should first of all `pip install Sphinx` inside your virtualenv. Then you can run `python setup.py build_sphinx`. This command will create documentation inside `build/sphinx/html/`. So run `firefox build/sphinx/html/index.html` and you can read it.

**See also:**

*Installation*

## 5.3 Documenting code

If you are a developer, you know that well-documented code is very important: it makes newcomers more comfortable hacking your project, it helps clarifying what's the goal of the code you are writing and how other parts of the project should use it. Keep in mind that libreant must be easily hackable, and the code should be kept reusable at all levels as much as possible.

Since 99% of libreant code is Python, we'll focus on it, and especially on python docstrings.

If you are writing a new module, or anyway creating a new file, the “module docstring” (that is, the docstring just at the start of the file) should explain what this module is useful for, which kind of objects will it contain, and clarify any possible caveat.

The same principle applies to classes and, to a lesser degree, to methods. If a class docstring is complete enough, it can be the case that function docstring is redundant. Even in that case, you should at least be very careful in giving meaningful names to function parameters: they help a lot, and come for free!

This chapter is dedicated to developers, and will guide you through code organization, design choices, etc. This is not a tutorial to python, nor to git. It will provide pointers and explanation, but will not teach you how to program.

## 6.1 Ingredients

libreant is coded in python2.7. Its main components are an [elasticsearch](#) db, a [Fsdb](#) and a web interface based on [Flask](#).

### 6.1.1 Details about libraries

Elasticsearch is a big beast. It has a lot of features and it can be scaring. We can suggest this [elasticsearch guide](#). The python library for elasticsearch, [elasticsearch-py](#), is quite simple to use, and has a nice documentation.

[Fsdb](#) is a quite simple “file database”: the main idea behind it is that it is a content-addressable storage. The address is simply the sha1sum of the content.

[Flask](#) is a “web microframework for python”. It’s not a big and complete solution like django, so you’ll probably get familiar with it quite soon.

## 6.2 Installation

### 6.2.1 Using virtualenv

We will assume that you are familiar with virtualenvs. If you are not, please get familiar!

Inside a clean virtualenv, run

```
python setup.py develop
```

You are now ready to develop. And you'll find two tools inside your `$PATH`: `webant` and `libreant-manage`. The first is a webserver that will run the web interface of libreant, while the second is a command-line tool to do basic operations with libreant: exporting/importing items, searching, etc.

## 6.2.2 Using Vagrant

Download, setup and run the virtual machine:

```
vagrant up
```

You will then find in `/liberant` the installation of liberant, you can login to the vagrant box with:

```
vagrant ssh
```

## 6.3 Code design

This section is devoted to get a better understanding on why the code is like it is, the principles that guides us, and things like that.

### 6.3.1 Design choices

**few assumptions about data** We try to be very generic about the items that libreant will store. We do not adhere to any standard about book catalogation, nor metadata organization, nor nothing like that. We leave the libraries free to set metadata how they prefer. There is only one mandatory field in items, which is `language`. The reason it is this way, is that it's important to know the language of the metadata in order for full-text search to work properly. There are also two somewhat-special fields: `title` and `actors`; they are not required, but are sometimes used in the code (being too much agnostic is soo difficult!)

**no big framework** we try to avoid huge frameworks like `django` or similar stuff. This is both a precise need, and a matter of taste. First of all, libreant uses many different storage resources (`elasticsearch`, `fsdb`, and this list will probably grow), so most frameworks will not fit our case. But it's also because we want to avoid that the code is "locked" in a framework and therefore difficult to fork.

### 6.3.2 File organization

`setup.py` is the file that defines how libreant is installed, how are packages built, etc. The most common reason you could care about it, is if you need to add some dependency to libreant.

#### libreantdb

`libreantdb/` is a package containing an abstraction over `elasticsearch`. Again: this is `elasticsearch-only`, and completely unaware of any other storage, or the logic of libreant itself.

#### webant

`webant/` is a package; you could think that it only contains web-specific logic, but this is not the case. Instead, all that is not in `libreantdb` is in `webant`, which is surely a bit counterintuitive.

The web application (defined in `webant.py`) “contains” a [Blueprint](#) called `agherant`. `Agherant` is the part of `libreant` that cares about “aggregating” multiple nodes in one single search engine. We believe that `agherant` is an important component, and if we really want to make `libreant` a distributed network, it should be very reusable. That’s why `agherant` is a blueprint: it should be reusable easily.

`manage.py` is what will be installed as `libreant-manage`: a simple command-line manager for lot of `libreant` operations. `libreant-manage` is meant to be a tool for developers (reproduce scenarios easily) and sysadmins (batch operations, debug), surely not for librarians! This program is actually based on [flask-script](#), so you may wonder why we use `flask` for something that is not web related at all; the point is that we use `flask` as an application framework more than a web framework.

`templates/` is... well, it contains templates. They are written with [jinja](#) templating language. The `render_template` function

## documentation

Documentation is kept on `doc/source/` and is comprised of `.rst` files. The syntax used is [restructuredText](#). Don’t forget to update documentation when you change something!

### 6.3.3 API

You can read [API](#)

### 6.3.4 Coding style

[PEP8](#) must be used in all the code.

Docstrings are used for autogenerating api documentation, so please don’t forget to provide clear, detailed explanation of what the module/class/function does, how to use it, when is it useful, etc. If you want to be really nice, consider using [restructured-text directives](#) to improve the structure of the documentation: they’re fun to use.

We care a lot about documentation, so please don’t leave documentation out-of-date. If you change the parameters that a function is accepting, please document it. If you are making changes to the end user’s experience, please fix the user manual.

Never put “binary” files in the source. With ‘binary’, we also mean “any files that could be obtained programmatically, instead of being included”. This is, for example, the case of `.mo`.

## 6.4 Testing

Unit tests are important both as a way of avoiding regressions and as a way to document how something behaves. If your code is testable, you should test it. Yes, even if its behaviour might seem obvious. If the code you are writing is not easy to test, you should think of making it more easy to test. We use [nose suite](#) to manage tests, you can run all the tests and read coverage summary by typing:

```
python setup.py test
```

**We usually follow these simple steps to add new tests:**

- create a directory named `test` inside the package you want to test
- create a file in this folder `test/test_sometestgroupname.py`
- write [test functions](#) inside this file

We prefer not to have one big file, instead we usually group tests in different file with a representative name. You can see a full testing example in the [preset package](#).

---

**Note:** if you are testing a new package remember to add the new package name in `cover-package` directive under `[nosetests]` section in `/setup.cfg` file.

---

## 6.5 Contributing

Like libreant? You can help!

We have a [bugtracker](#), and you are welcome to pick tasks from there :) We use it also for discussions. Our most typical way of proposing patches is to open a pull request on github; if, for whatever reason, you are not comfortable with that, you can just contact us by email and send a patch, or give a link to your git repository.



## 7.1 archivant package

### 7.1.1 Submodules

## 7.2 conf package

### 7.2.1 Submodules

`conf.config_utils.from_envvar_file` (*envvar*, *environ=None*)

`conf.config_utils.from_envvars` (*prefix=None*, *environ=None*, *envvars=None*, *as\_json=True*)

Load environment variables in a dictionary

Values are parsed as JSON. If parsing fails with a `ValueError`, values are instead used as verbatim strings.

#### Parameters

- **prefix** – If `None` is passed as *envvars*, all variables from *environ* starting with this prefix are imported. The prefix is stripped upon import.
- **envvars** – A dictionary of mappings of environment-variable-names to Flask configuration names. If a list is passed instead, names are mapped 1:1. If `None`, see *prefix* argument.
- **environ** – use this dictionary instead of `os.environ`; this is here mostly for mockability
- **as\_json** – If `False`, values will not be parsed as JSON first.

`conf.config_utils.from_file` (*fname*)

`conf.config_utils.load_configs` (*envvar\_prefix*, *path=None*)

Load configuration

The following steps will be undertake:

- It will attempt to load configs from file: if *path* is provided, it will be used, otherwise the path will be taken from envvar *envvar\_prefix* + “SETTINGS”.
- all envvars starting with *envvar\_prefix* will be loaded.

## 7.3 libreantdb package

### 7.3.1 Submodules

## 7.4 presets package

**class** `presets.PresetManager` (*paths*, *strict=False*)  
Bases: `object`

PresetManager deals with presets loading, validating, storing

you can use it like this:

```
pm = PresetManager(["/path/to/presets/folder", "/another/path"])
```

**MAX\_DEPTH** = 5

### 7.4.1 Submodules

**class** `presets.presetManager.Preset` (*body*)  
Bases: `presets.presetManager.Schema`

A preset is a set of rules and properties denoting a class of object

**Example:** A preset could be used to describe which properties an object that describe a book must have. (title, authors, etc)

**check\_id** ()

**fields** = {'allow\_upload': {'default': True, 'required': False, 'type': <type 'bool'>}, 'description': {'default': '', 'required': True}}

**validate** (*data*)

Checks if *data* respects this preset specification

It will check that every required property is present and for every property type it will make some specific control.

**exception** `presets.presetManager.PresetException` (*message*)  
Bases: `exceptions.Exception`

**exception** `presets.presetManager.PresetFieldTypeException` (*message*)  
Bases: `presets.presetManager.PresetException`

**class** `presets.presetManager.PresetManager` (*paths*, *strict=False*)  
Bases: `object`

PresetManager deals with presets loading, validating, storing

you can use it like this:

```
pm = PresetManager(["/path/to/presets/folder", "/another/path"])
```

**MAX\_DEPTH** = 5

**exception** `presets.presetManager.PresetMissingFieldException` (*message*)

Bases: `presets.presetManager.PresetException`

**class** `presets.presetManager.Property` (*body*)

Bases: `presets.presetManager.Schema`

A property describe the format of a peculiarity of a preset

**check\_id**()

**check\_type**()

**check\_values**()

**fields** = {'values': {'required': 'required\_values', 'type': <type 'list'>, 'check': 'check\_values'}, 'required': {'default':

**required\_values**()

**types** = ['string', 'enum']

*fields* is used as in Preset class

**class** `presets.presetManager.Schema`

Bases: `object`

Schema is the parent of all the classes that needs to verify a specific object structure.

**all child class in order to use schema validation must:**

- describe the desired object schema using *self.fields*
- save input object in *self.body*

*self.fields* must be a dict, where keys match the relative *self.body* keys and values describe how relative *self.body* valuse must be.

Example:

```
self.fields = { 'description': {
    'type': basestring,
    'required': False,
    'default': ""
},
    'allow_upload': {
    'type': bool,
    'required': False,
    'default': True
    }
}
```

**fields** = {}

## 7.5 users package

### 7.5.1 Submodules



### 8.1 next

#### 8.1.1 Dependencies:

- Added support for Elasticsearch 5.x and 6.x (PR #321)
- Gevent: added support for gevent version till 1.2.2

### 8.2 0.5

- Added support to Elasticsearch 2.x versions. (PR #281)
- Changed default capability for anonymous (non logged) user: now she can read all volumes in the collection.

Tip: if you have an existing and already initialized user database, it won't be changed, i.e. if you upgrade from a previous version of libreant and you have existing users, the anonymous user won't get the read capability. In the case you want to add this capability to the already existing anonymous user you can use the following command:

```
libreant-users --users-db <users-db-url> group cap-add anonymous "volumes/*" R
```

#### 8.2.1 CLI:

- Added new command *libreant-db import* to import volumes all at once. (PR #291)

#### 8.2.2 Web Interface:

- While adding a new book if it is available the language will be autocompleted using the one suggested by the client's browser. ( based on *Accept-Language* http field). Thanks @leonaard (PR #288)

### 8.2.3 API:

- **Added endpoints to retrieve collections**
  - `/api/v1/groups/`
  - `/api/v1/users/`
  - `/api/v1/capabilities/`

### 8.2.4 Dependencies:

- Fsdb: added support till version *1.2.1* (PR #277)
- Gevent: added support for the new version *1.1.1* (PR #298)
- Flask: added support till version *0.11.1* (PR #299)

### 8.2.5 Bugfixes:

- **#255 Libreant starts also if it fails to read the conf file:** fixed by PR #260. If some error is encountered while reading the configuration file the stack trace will be printed if the debug mode is active otherwise a colored one-line message with the cause of the error will be printed. Moreover the path of the configuration file will be printed if available.
- **#283 Read configuration file error:** If the configuration is a valid JSON formatted file but it's not a dictionary an exception is raised. (PR #286)
- Tests for Webant were leaving leftover files around ( commit 1c050a8 )
- In single-user-mode all the users related REST api endpoints are disabled (PR #278)
- CLI: don't print 'Error' string twice (PR #279)
- **#287 missing authentication/authorization layer for REST api.** For the moment the only supported authentication method is the cookie based one ( login through the web UI )

## 8.3 0.4

### 8.3.1 Web Interface:

- The page to modify metadata of volumes has been added. If you have enough permission you should see a button with a pencil on the single-volume-view page.
- Added support for paginated results in search page.

### 8.3.2 CLI:

- added new command `libreant-db insert-volume` to insert a volume along with its attachments.
- added new command `libreant-db attach` to attach new files to an already existing volume.

### 8.3.3 Logs:

- changed default log level to INFO.
- all startup messages are now printed using loggers.
- using recent versions of gevent ( $\geq 1.1b1$ ) it is now possible to have a completely uniform log format.

### 8.3.4 Warning:

- Due to breaking changes introduced in new version of Elasticsearch (deprecation of `_timestamp` field), it is not possible to use libreant with version of Elasticsearch major or equal to 2.0. Probably in the next release we'll provide support for these versions.

## 8.4 0.3

### 8.4.1 Major changes:

- Implemented a role-based access control layer. This means that libreant now support the common login procedure. This functionality isn't documented yet, anyway you can use the brand new `libreant-users` command to manage users, groups and capabilities, and enable this feature at runtime with the `--users-db` parameter. The default user is (user: admin, password: admin)

### 8.4.2 Web interface:

- Added possibility to delete a volume through a button on the single-volume-view page.
- New user menu (only in users-mode)
- New login/logout pages.
- Improved error messages/pages

### 8.4.3 Deployment:

- Removed elasticsearch strong dependency. Now libreant can be started with elasticsearch still not ready or not running.
- Bugfix: make libreant command exits with code 1 on exception.
- Fixed `elasticsearch-py` version dependency. Now the version must be  $\geq 1$  and  $< 2$ .
- Reloader is used only in debug mode (`--debug`).
- More uniform logs.

### 8.4.4 Documentation:

- The suggested version for elasticsearch installation has been updated: 1.4 -> 1.7
- A lot of packages have been inserted in the official docs.





## CHAPTER 9

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### **c**

`conf`, [21](#)

`conf.config_utils`, [21](#)

### **p**

`presets`, [22](#)

`presets.presetManager`, [22](#)



## C

`check_id()` (presets.presetManager.Preset method), [22](#)  
`check_id()` (presets.presetManager.Property method), [23](#)  
`check_type()` (presets.presetManager.Property method),  
[23](#)  
`check_values()` (presets.presetManager.Property method),  
[23](#)  
`conf` (module), [21](#)  
`conf.config_utils` (module), [21](#)

## F

`fields` (presets.presetManager.Preset attribute), [22](#)  
`fields` (presets.presetManager.Property attribute), [23](#)  
`fields` (presets.presetManager.Schema attribute), [23](#)  
`from_envvar_file()` (in module `conf.config_utils`), [21](#)  
`from_envvars()` (in module `conf.config_utils`), [21](#)  
`from_file()` (in module `conf.config_utils`), [21](#)

## L

`load_configs()` (in module `conf.config_utils`), [21](#)

## M

`MAX_DEPTH` (presets.PresetManager attribute), [22](#)  
`MAX_DEPTH` (presets.presetManager.PresetManager attribute), [22](#)

## P

`Preset` (class in `presets.presetManager`), [22](#)  
`PresetException`, [22](#)  
`PresetFieldTypeException`, [22](#)  
`PresetManager` (class in `presets`), [22](#)  
`PresetManager` (class in `presets.presetManager`), [22](#)  
`PresetMissingFieldException`, [22](#)  
`presets` (module), [22](#)  
`presets.presetManager` (module), [22](#)  
`Property` (class in `presets.presetManager`), [23](#)

## R

`required_values()` (presets.presetManager.Property method), [23](#)

## S

`Schema` (class in `presets.presetManager`), [23](#)

## T

`types` (presets.presetManager.Property attribute), [23](#)

## V

`validate()` (presets.presetManager.Preset method), [22](#)