

---

# Burp-UI Documentation

*Release 0.6.0*

**Ziirish**

**May 14, 2018**



---

## Contents

---

<b>1</b>	<b>Documentation</b>	<b>3</b>
1.1	Introduction . . . . .	3
1.2	Requirements . . . . .	4
1.3	Architecture . . . . .	6
1.4	Installation . . . . .	7
1.5	Basic usage . . . . .	8
1.6	Running . . . . .	10
1.7	Upgrading . . . . .	11
1.8	Advanced usage . . . . .	12
1.9	Manage . . . . .	24
1.10	Celery . . . . .	30
1.11	WebSocket . . . . .	31
1.12	Gunicorn . . . . .	32
1.13	Docker . . . . .	37
1.14	bui-agent . . . . .	39
1.15	Service . . . . .	42
1.16	Contributing . . . . .	43
1.17	Changelog . . . . .	46
1.18	FAQ . . . . .	54
1.19	Step By Step . . . . .	57
1.20	Developer Guide . . . . .	61
	<b>HTTP Routing Table</b>	<b>111</b>



Burp-UI is a web-ui for [Burp](#) backup written in python with Flask and jQuery/Bootstrap. You may have a look at the [README](#) file first. There is also a dedicated [FAQ](#).



## 1.1 Introduction

**Burp-UI** is a web-based interface for **Burp**. Its purpose is to give you a *nice* way to monitor your backups with some dashboards, but you will also have the ability to download files from backups and to configure your burp-server.

The project also provides a fully documented **API** so that you can develop any front-end you like on top of it. The core will take care of the communication with the burp server(s) for you.

**Note:** Although the **Burp**'s author and I exchange a lot, our products are totally distinct. So I would like people to understand some issues might be related to **Burp-UI**, but some other might be related to **Burp** and I may not be able to help you in the later case. There is a dedicated mailing-list for **Burp** related issues. You can find details [here](#)

### 1.1.1 Compatibility

Burp version Backend	1	2
< 1.3.48		
1.3.48 => 1.4.40	X	
2.0.0 => 2.0.16		
2.0.18 => 2.0.X protocol 1		X
2.0.18 => 2.0.X protocol 2		X*

\* The protocol 2 is in heavy development Burp side so the support in **Burp-UI** is best effort and all features (such as server-initiated restoration) are not available.

### 1.1.2 Known Issues

Because it's an Open Source project, people are free (and encouraged) to open issues in the [bug-tracker](#). You will also find there the current opened issues.

There are also a few issues unrelated to the code itself:

### 1. SSH issue

People that would like to clone the repository over SSH will face an authentication failure even if they added a valid SSH key in their user settings. The reason is I only have *one* public IP address so I must use port redirections to have multiple SSH instances running. To fix the issue, you should configure your SSH client by adding the following lines in your `~/.ssh/config` file:

```
Host git.ziirish.me
  Port 2222
```

## 1.2 Requirements

Please note that, **Burp-UI** must be running on the same server that runs the `burp-server` for some features.

---

**Note:** At the moment, **Burp-UI** and this doc is mostly debian-centric but feel free to contribute for other distributions!

---

### 1.2.1 Python

**Burp-UI** is built against python 2.7. The support for python 2.6 has been removed since it is not supported anymore by the CPython core team. Unit tests are ran against python 2.7 and python 3.6. If you encounter compilation errors with one of these version, feel free to report them.

### 1.2.2 Libraries

Some libraries are required to be able to compile some requirements:

```
apt-get install libffi-dev libssl-dev python-dev python-pip
```

### 1.2.3 LDAP

For LDAP authentication (optional), we need extra dependencies. You can install them using the following command:

```
pip install "burp-ui[ldap_authentication]"
```

### 1.2.4 Redis

If you wish to use redis for Caching and/or managing user sessions, you need additional dependencies:

```
pip install "burp-ui[gunicorn-extra]"
```

### 1.2.5 Celery

The celery worker also needs additional dependencies that you can install using:



```
pip install "burp-ui[celery]"
```

### 1.2.6 SQL

If you need persistent data, you will need additional dependencies as well:

```
pip install "burp-ui[sql]"
```

Now if you want to use a MySQL database, you will need the proper driver. For instance:

```
pip install mysqlclient
```

**Warning:** The MySQL driver does not seem to play nicely with concurrency, you should set `preload=False` within your gunicorn config.

To use a PostgreSQL database, you need the `psycopg2` driver:

```
pip install psycopg2
```

**Warning:** The PostgreSQL driver does not seem to play nicely with concurrency, you should set `preload=False` within your gunicorn config.

### 1.2.7 Limiter

If you want to rate-limit the API, you will need additional dependencies too:

```
pip install flask-limiter
```

### 1.2.8 WebSocket

If you want to enable the WebSockets support, you need to install the following:

```
pip install "burp-ui[websocket]"
```

### 1.2.9 Burp1

The `burp1` backend supports burp versions from 1.3.48 to 1.4.40. With these versions of burp, the status port is only listening on the local machine loopback interface (ie. `localhost` or `127.0.0.1`). It means you *MUST* run **Burp-UI** on the same host that is running your burp server in order to be able to access burp's statistics. Alternatively, you can use a `bui-agent`.

### 1.2.10 Burp2

The `burp2` backend supports only burp 2.0.18 and above. Some versions are known to contain critical issues resulting in a non-functional **Burp-UI**: 2.0.24, 2.0.26 and 2.0.30 If you are using an older version of burp2 **Burp-UI** will fail to start.

### 1.2.11 Getting started

The first thing to do before digging into [Burp-UI](#) is probably to read its [architecture](#) in order to understand how it works. Once it's done, you can refer to the [installation](#) page.

## 1.3 Architecture

This section is a must-read in order to understand what is going on between [Burp](#) and [Burp-UI](#).

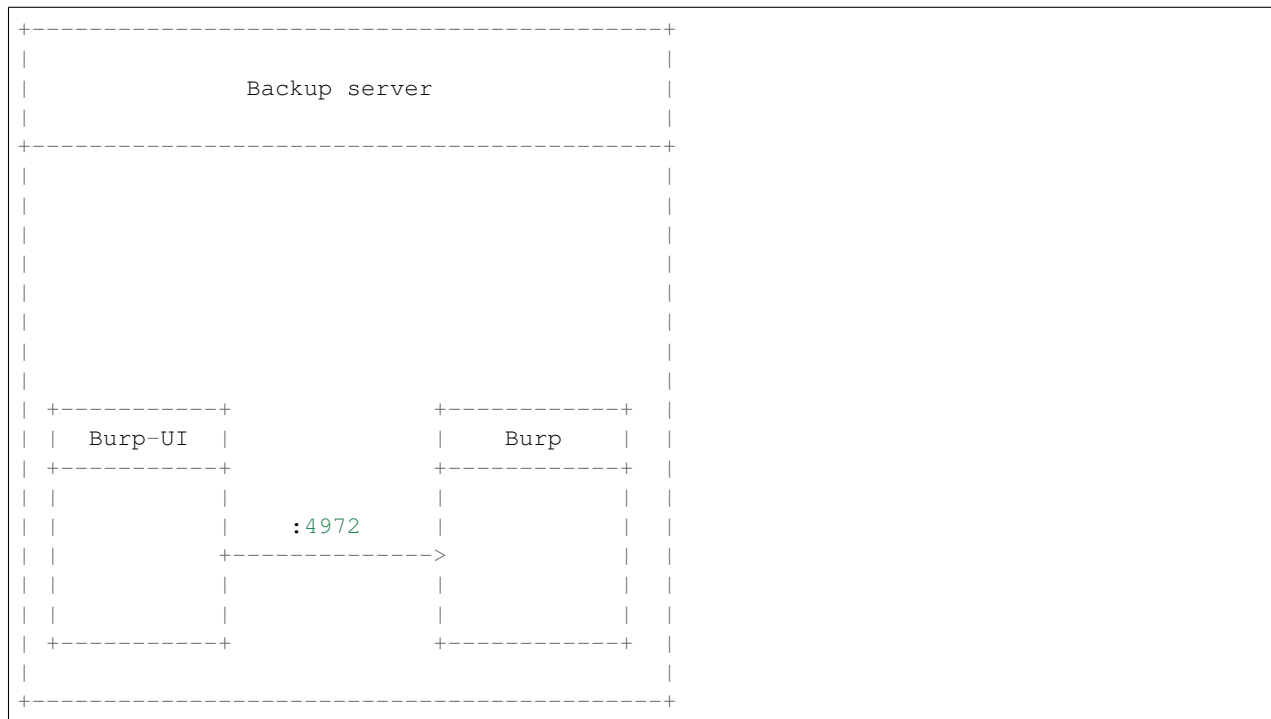
Both projects are lead by two different people so please report your issues to the right project.

The *Burp1* and *Burp2* backends behave slightly differently due to some changes in the core or [Burp](#). You can refer to the [Burp's documentation](#) for details.

### 1.3.1 Burp 1.x

If you are running [Burp 1.x](#), you **MUST** install [Burp-UI](#) on the same host (or at least setup a [bui-agent](#) locally). This limitation is due to the fact burp 1.x only exposes its *status port* (the port 4972 by default) to localhost (either 127.0.0.1 or ::1). [Burp-UI](#) then just opens a connexion to `localhost : 4972` in order to communicate with [Burp](#).

Here is a little illustration:



### 1.3.2 Burp 2.x

If you are running [Burp 2.x](#), you can host [Burp-UI](#) on a different server, but I don't recommend it if you wish to be able to use all the features.

The [Burp 2.x status port](#) has been completely reworked. It can now be published to remote hosts, but the *status protocol* is not compatible with 1.x.

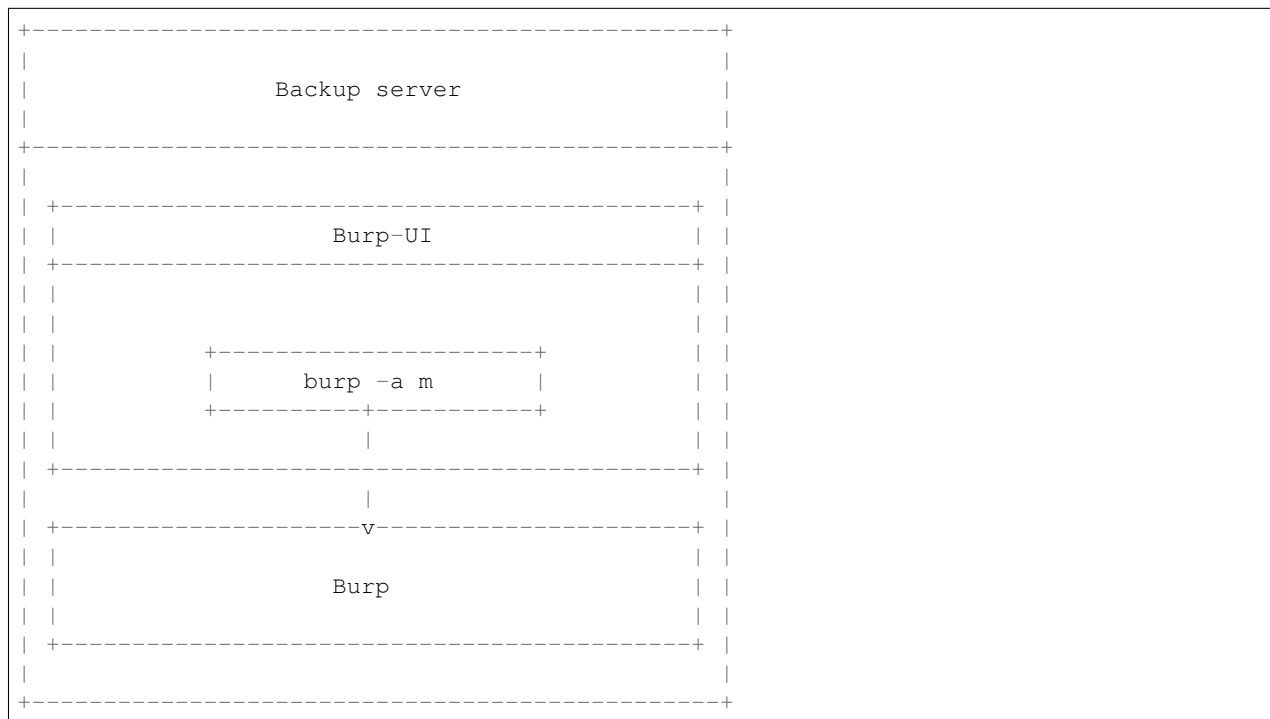
The *status port* is now accessed through a **Burp** client thanks to the `-a m` flag. The client will then take care to open the connexion with the server securing the communication with SSL. The **Burp** server also supports basic *ACL* though the `restore_client` option.

By default, a client will only be able to view its own reports/stats. If you want to be able to monitor other clients, you need to be added as a `restore_client` for those clients (this can be done by editing the client configuration file in the `clientconfdir` directory on the server). Alternatively, you can add this option in the `burp-server.conf` file for this setting to be applied globally for all the clients.

Ok, now why did I tell you all this? Well, because the **Burp-UI's** backend for **Burp 2.x** is basically just a wrapper around `burp -a m`.

It means that when you start **Burp-UI** with `version = 2`, the command `<burpbin> -c <bconfcli> -a m` (where `<burpbin>` defaults to `/usr/sbin/burp` and `<bconfcli>` defaults to `/etc/burp/burpui.cfg`). Of course this command will be ran with the same permissions and privileges as **Burp-UI** itself.

And here is a little illustration to summarize all this:



## 1.4 Installation

Burp-UI is written in Python with the [Flask](#) micro-framework. The easiest way to install Burp-UI is to use `pip`.

```
pip install burp-ui
```

You can setup various parameters in the `burpui.cfg` file. This file can be specified with the `-c` flag or should be present in `/etc/burp/burpui.cfg`. By default **Burp-UI** ships with a sample file located in `$INSTALLDIR/share/burpui/etc/burpui.sample.cfg`. (`$INSTALLDIR` defaults to `/usr/local` when using **pip** **outside** a virtualenv)

**Note:** It is advised to copy the sample configuration in `/etc/burp/burpui.cfg` and to edit this file so that it is not overwritten on every upgrade.

Then you can run `burp-ui: burp-ui`

By default, `burp-ui` listens on localhost on port 5000.

You can then point your browser to <http://127.0.0.1:5000/>

### 1.4.1 Upgrade

In order to upgrade **Burp-UI** to the latest stable version, you can run the following command:

```
pip install --upgrade burp-ui
```

---

**Note:** If you encounter any issue after upgrading to the latest stable release, make sure you read the [upgrading](#) page.

---

## 1.5 Basic usage

### 1.5.1 General Instructions

#### Restoration

**Burp-UI** tries to be as less intrusive as possible with **Burp** internals. In order to make the *online* restoration/download functionality work, you need to check a few things:

1. Provide the full path of the burp (client) binary file (field *burpbin* in **burp-ui** configuration)
2. Provide a burp-client configuration file (field *bconfcli* in **burp-ui** configuration)
3. Provide the full path of an empty directory where a temporary restoration will be made. This involves you have enough space left on that location on the server that runs **Burp-UI**
4. Launch **Burp-UI** with a user that can proceed restorations and that can write in the directory mentioned above
5. Make sure the client provided in 2. can restore files of other clients (option *restore\_client* in burp-server configuration). The *restore\_client* is the *cname* you provided in your client configuration file (see 2.)

#### Burp 2

When using the **burp2** backend, **Burp-UI** can be executed on any machine as long as you can access the burp status port, but you will not be able to edit the burp server configuration file within the *settings* view of **Burp-UI**. You also need to configure a *restore\_client* on your burp server corresponding to the client you will use through **Burp-UI** (see the *restoration procedure* above for details on how to do this)

### 1.5.2 Tools

Since *v0.5.0*, **Burp-UI** ships with some tools to help you configure both Burp and **Burp-UI**. Please refer to the [bui-manage](#) page for details.

### 1.5.3 Options

```
usage: burp-ui [-h] [-v] [-d] [-V] [-c <CONFIG>] [-l <FILE>]
              [-i <MIGRATIONS_DIR>]
              ...

positional arguments:
  remaining

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         increase output verbosity (e.g., -vv is more verbose
                        than -v)
  -d, --debug          enable debug mode
  -V, --version         print version and exit
  -c <CONFIG>, --config <CONFIG>
                        burp-ui configuration file
  -l <FILE>, --logfile <FILE>
                        output logs in defined file
  -i <MIGRATIONS_DIR>, --migrations <MIGRATIONS_DIR>
                        migrations directory
```

**Note:** Since v0.4.0, the *bind* and *port* options are not supported anymore. You can still force the port and listening address as explained below or you can have a look at the [upgrading](#) section of the documentation for further details.

### 1.5.4 Developer options

Since v0.4.0, **Burp-UI** uses the new Flask's *CLI* module. This change brings new options to help you debug your development environment:

```
Usage: flask run [OPTIONS]

Runs a local development server for the Flask application.

This local server is recommended for development purposes only but it can
also be used for simple intranet deployments. By default it will not
support any sort of concurrency at all to simplify debugging. This can be
changed with the --with-threads option which will enable basic
multithreading.

The reloader and debugger are by default enabled if the debug flag of
Flask is enabled and disabled otherwise.

Options:
  -h, --host TEXT          The interface to bind to.
  -p, --port INTEGER       The port to bind to.
  --reload / --no-reload   Enable or disable the reloader. By default
                           the reloader is active if debug is enabled.
  --debugger / --no-debugger
                           Enable or disable the debugger. By default
                           the debugger is active if debug is enabled.
  --eager-loading / --lazy-loader
                           Enable or disable eager loading. By default
                           eager loading is enabled if the reloader is
                           disabled.
```

(continues on next page)

(continued from previous page)

```
--with-threads / --without-threads    Enable or disable multithreading.  
--help                               Show this message and exit.
```

Some options are redundant with [Burp-UI](#)'s one.

### 1.5.5 Example

By default, the embedded server listens on *127.0.0.1* on port *5000*, if you wish to change this, you would run something like:

```
burp-ui -- -h 0.0.0.0 -p 8080
```

---

**Note:** The `--` is mandatory.

---

## 1.6 Running

There are several ways to run [Burp-UI](#). You can either use the embedded Flask's [development server](#) or you can use any of the [deployment options](#) provided by Flask.

---

**Note:** I personally focus on [gunicorn](#) support for production deployments

---

### 1.6.1 Sandboxing

If you want to play with [Burp-UI](#) to *PoC* it or if you are going to be the only user, you can absolutely use the embedded server. If you plan to run [Burp-UI](#) in production, then you should go with [Gunicorn](#).

#### Option 1

You can run the embedded server with the following command:

```
burp-ui
```

By default, the server listens on *localhost:5000*. You can easily change this by adding the `-- -h x.x.x.x -p yyyy` options. See [here](#) for details.

#### Option 2

Prior to *v0.4.0*, you could specify the *bind* and *port* option within the [Burp-UI](#) configuration file. You can still use this behavior by running:

```
python -m burpui -m legacy [--help]
```

---

**Note:** Since *v0.5.0*, you can also use the `burp-ui-legacy` command to achieve the same thing.

---

## 1.6.2 Production

Like I said earlier, I recommend using [Gunicorn](#) for production deployments. You can refer to the dedicated [gunicorn](#) page of this documentation to know everything on how to [Burp-UI](#) through [Gunicorn](#).

## 1.6.3 Going further

Please refer to the [advanced usage](#) page for details on how to use/customize [Burp-UI](#).

# 1.7 Upgrading

This page is here to help you upgrading from previous versions of [Burp-UI](#) to the latest version. Each section presents major/breaking changes, new requirements and new options. For a complete list of changes, you may refer to the [CHANGELOG](#) page.

### 1.7.1 v0.6.0

- **Breaking** - The *BASIC* ACL engine will now grant users on all agents if they are not explicitly defined. It means that if you have a user called `example1` with two agents (burp servers in multi-agent mode) on which you have respectively two clients called `example1`, the user `example1` will be granted on both clients on the two agents. You can disable this behavior with the `legacy` option. See the [BASIC ACL](#) documentation for details.
- **Breaking** - A new `[ACL]` section has been created in order to control the new ACL engine behavior. See the [ACL](#) documentation for details.
- **Breaking** - The *Burp1* and *Burp2* configuration sections have been merged into one single *Burp* section. See the [Versions](#) documentation for details.
- **Breaking** - The *running* backups are now displayed in green instead of blue.
- **Breaking** - The docker postgresql image was upgraded from 9.6 to 10.1, you'll have to manually upgrade/migrate your data [following this documentation](#).
- **Breaking** - The `docker-compose.yml` file now uses the `version: '2'` format.
- **Breaking** - The old config file format with colons (`:`) as separator is no more supported.
- **New** - Plugin system to enhance ACL and Authentication backends. See the [Plugins](#) documentation for details.
- **New** - WebSocket support for better/smarter notifications.
- **New** - Client configuration templates.
- **New** - Administration panel to manage both Authentications and Authorizations.

### 1.7.2 v0.5.0

- **Breaking** - The *standalone* option has been renamed to *single* to avoid confusion.
- **Breaking** - The *bui-agent* has now its own independent package to reduce dependencies, you can install it with the `pip install burp-ui-agent` command. Alternatively, there is now a `bui-agent-legacy` command provided by the `burp-ui` package.
- **Breaking** - The database schema evolved between *v0.4.0* and *v0.5.0*. In order to apply these modifications, you **MUST** run the `bui-manage db upgrade` command before restarting your [Burp-UI](#) application (if you are using celery, you must restart it too).

- **New** - The `bui-manage` tool brings two new commands:
  - `diag` whose documentation is available [here](#)
  - `sysinfo` whose documentation is available [here](#)

More details on the [Manage](#) and [Celery](#) pages.

### 1.7.3 v0.4.0

- **Breaking** - Due to the use of the new Flask's embedded server, it is no longer possible to serve the application over SSL (HTTPS) anymore from within the Flask's server. You'll need to use a dedicated application server for this purpose such as [gunicorn](#) or a reverse-proxy. The `bind` and `port` option have also been removed due to the same reason.

Or you can use the `python -m burpui -m legacy [--help]` command that **SHOULD** be backward compatible (but note that no further support will be provided since it is not the Flask's default behavior anymore).

- **Breaking** - The database schema evolved between `v0.3.0` and `v0.4.0`. In order to apply these modifications, you **MUST** run the `bui-manage db upgrade` command before restarting your [Burp-UI](#) application (if you are using celery, you must restart it too).

More details on the [Manage](#) and [Celery](#) pages.

- **Breaking** - Plain text passwords are deprecated since `v0.3.0` and are now disabled by default. It means you should not manually add new users in your burp-ui configuration anymore with `login = password` but you should now use the `bui-manage` command instead.
- **Breaking** - The default `version` setting has been set to 2 instead of 1 since burp-2.0.54 is now the stable release.
- **New** - The `bui-manage` tool can now help you setup both [Burp](#) and [Burp-UI](#).
- **New** - The SQL requirements have evolved, you **MUST** run `pip install --upgrade "burp-ui[sql]"` if you wish to keep using persistent storage.

### 1.7.4 v0.3.0

- **New** - `bui-manage` tool: This tool is used to setup database (see [Manage](#)).
- **New** - `bui-celery` tool: This tool is used to run a celery runner (see [Celery](#)).
- **Breaking** - Configuration file format changed. Colons (:) must be replaced by equals (=). Besides, some settings containing spaces should be surrounded by quotes. \*Note\*: The conversion is mostly automatic, but you should keep an eye on it though.
- **New** - Basic authentication backend now supports hashed passwords (*Note*: plain text passwords are now deprecated and the support will be dropped in `v0.4.0`). You can create new users with the `bui-manage` tool, passwords generated through this tool are hashed. *Note*: Starting with `v0.4.0`, plain text passwords will be automatically hashed.
- **New** - Local authentication backend allows you to login using local accounts through pam.

## 1.8 Advanced usage

[Burp-UI](#) has been written with modularity in mind. The aim is to support [Burp](#) from the stable to the latest versions. [Burp](#) exists in two major versions: 1.x.x and 2.x.x.



Both *Versions* are supported by **Burp-UI** thanks to its modular design. The consequence is you have various options in the configuration file to suite everybody needs.

There are also different modules to support *Authentication* and *ACL* within the web-interface.

**Warning:** **Burp-UI** tries to be as less intrusive as possible, nevertheless it ships with the ability to manage **Burp**'s configuration files. This feature **requires** **Burp-UI** to be launched on the **same** server that hosts your **Burp** instance. You also have to make sure the user that runs **Burp-UI** has **enough** privileges to edit those files.

## 1.8.1 Configuration

The `burpui.cfg` configuration file contains a `[Global]` section as follow:

```
[Global]
# burp server version 1 or 2
version = 2
# Handle multiple bui-servers or not
# If set to 'false', you will need to declare at least one 'Agent' section (see
# below)
single = true
# authentication plugin (mandatory)
# list the misc/auth directory to see the available backends
# to disable authentication you can set "auth = none"
# you can also chain multiple backends. Example: "auth = ldap,basic"
# the order will be respected unless you manually set a higher backend priority
auth = basic
# acl plugin
# list misc/acl directory to see the available backends
# default is no ACL
acl = basic
# You can change the prefix if you are behind a reverse-proxy under a custom
# root path. For example: /burpui
# You can also configure your reverse-proxy to announce the prefix through the
# 'X-Script-Name' header. In this case, the bellow prefix will be ignored in
# favour of the one announced by your reverse-proxy
prefix = none
# list of paths to look for external plugins
plugins = none
```

Each option is commented, but here is a more detailed documentation:

- *version*: What version of **Burp** this **Burp-UI** instance manages. Can either be *1* or *2*. This parameter determines which backend is loaded at runtime.

(see *Versions* for more details)

- *single*: **Burp-UI** can run in two different modes. If it runs in single mode (meaning you set this parameter to *true*), you can only address **one** **Burp** server of the version specified by the previous parameter.

If this option is set to *false*, **Burp-UI** will run as a *proxy* allowing you to address multiple **Burp** servers. In this mode, you need to configure **at least one** *Agent* section in your configuration file. You also need to run one `bui-agent` per server.

(see *Modes* for more details)

- *auth*: What *Authentication* backend to use.
- *acl*: What *ACL* module to use.

- *prefix*: You can host **Burp-UI** behind a sub-root path. See the [gunicorn](#) page for details.
- *plugins*: Specify a list of paths to look for external plugins. See the [Plugins](#) page for details on how to write plugins.

There is also a [UI] section in which you can configure some *UI* parameters:

```
[UI]
# refresh interval of the pages in seconds
refresh = 180
# refresh interval of the live-monitoring page in seconds
liverefresh = 5
# list of labels to ignore (you can use regex)
ignore_labels = "color:.*", "custom:.*"
# format label using sed-like syntax
format_labels = "s/^os:\s*//"
# default strip leading path value for file restorations
default_strip = 0
```

Each option is commented, but here is a more detailed documentation:

- *refresh*: Time in seconds between two refresh of the interface.
- *liverefresh*: Time in seconds between two refresh of the *live-monitor* page.
- *ignore\_labels*: List of labels to ignore from parsing (regex are supported).
- *format\_labels*: List of *sed-like* expressions to transform labels. Example: "s/^os:\s\*//", "s/i/o/" will transform the label `os: Windows` into `Wondows`.
- *default\_strip*: Number of leading paths to strip by default while restoring files.

## 1.8.2 Production

The `burpui.cfg` configuration file contains a [Production] section as follow:

```
[Production]
# storage backend for session and cache
# may be either 'default' or 'redis'
storage = default
# session database to use
# may also be a backend url like: redis://localhost:6379/0
# if set to 'redis', the backend url defaults to:
# redis://<redis_host>:<redis_port>/0
# where <redis_host> is the host part, and <redis_port> is the port part of
# the below "redis" setting
session = default
# cache database to use
# may also be a backend url like: redis://localhost:6379/0
# if set to 'redis', the backend url defaults to:
# redis://<redis_host>:<redis_port>/1
# where <redis_host> is the host part, and <redis_port> is the port part of
# the below "redis" setting
cache = default
# redis server to connect to
redis = localhost:6379
# whether to use celery or not
# may also be a broker url like: redis://localhost:6379/0
# if set to "true", the broker url defaults to:
```

(continues on next page)

(continued from previous page)

```
# redis://<redis_host>:<redis_port>/2
# where <redis_host> is the host part, and <redis_port> is the port part of
# the above "redis" setting
celery = false
# database url to store some persistent data
# none or a connect string supported by SQLAlchemy:
# http://docs.sqlalchemy.org/en/latest/core/engines.html#database-urls
# example: sqlite:///var/lib/burpui/store.db
database = none
# whether to rate limit the API or not
# may also be a redis url like: redis://localhost:6379/0
# if set to "true" or "redis" or "default", the url defaults to:
# redis://<redis_host>:<redis_port>/3
# where <redis_host> is the host part, and <redis_port> is the port part of
# the above "redis" setting
# Note: the limiter only applies to the API routes
limiter = false
# limiter ratio
# see https://flask-limiter.readthedocs.io/en/stable/#ratelimit-string
ratio = 60/minute
```

### 1.8.3 WebSocket

The [WebSocket] section defines specific options for the WebSocket server. You will find details on how to use this feature in the [WebSocket](#) page.

```
[WebSocket]
## This section contains WebSocket server specific options.
# whether to enable websocket or not
enabled = true
# whether to embed the websocket server or not
# if set to "true", you should have only *one* unicorn worker
# see here for details:
# https://flask-socketio.readthedocs.io/en/latest/#unicorn-web-server
embedded = false
# what broker to use to interact between websocket servers
# may be a redis url like: redis://localhost:6379/0
# if set to "true" or "redis" or "default", the url defaults to:
# redis://<redis_host>:<redis_port>/4
# where <redis_host> is the host part, and <redis_port> is the port part of
# the above "redis" setting
# set this to none to disable the broker
broker = redis
# if you choose to run a dedicated websocket server (with embedded = false)
# you can specify here the websocket url. You'll need to double quote your
# string though.
# example:
# url = "document.domain + ':5001'"
url = none
# whether to enable verbose websocket server logs or not (for development)
debug = false
```

## 1.8.4 Experimental

There is a [Experimental] section for features that have not been deeply tested:

```
[Experimental]
## This section contains some experimental features that have not been deeply
## tested yet
# enable zip64 feature. Python doc says:
# « ZIP64 extensions are disabled by default because the default zip and unzip
# commands on Unix (the InfoZIP utilities) don't support these extensions. »
zip64 = false
```

These options are also available in the `bui-agent` configuration file.

## 1.8.5 Security

The [Security] section contains options to harden the security of the application:

```
[Security]
## This section contains some security options. Make sure you understand the
## security implications before changing these.
# list of 'root' paths allowed when sourcing files in the configuration.
# Set this to 'none' if you don't want any restrictions, keeping in mind this
# can lead to accessing sensible files. Defaults to '/etc/burp'.
# Note: you can have several paths separated by comas.
# Example: /etc/burp,/etc/burp.d
includes = /etc/burp
# if files already included in config do not respect the above restriction, we
# prune them
enforce = false
# enable certificates revocation
revoke = false
# remember_cookie duration in days
cookietime = 14
# whether to use a secure cookie for https or not. If set to false, cookies
# won't have the 'secure' flag.
# This setting is only useful when HTTPS is detected
scookie = true
# application secret to secure cookies. If you don't set anything, the default
# value is 'random' which will generate a new secret after every restart of your
# application. You can also set it to 'none' although this is not recommended.
appsecret = random
```

Some of these options are also available in the `bui-agent` configuration file.

## 1.8.6 Modes

Burp-UI provides two modes:

- *Single*
- *Multi-Agent*

These modes allow you to either access a single `Burp` server or multiple `Burp` servers hosted on separated hosts.

## Single

This mode is the **default** and the easiest one. It can be activated by setting the *single* parameter in the [Global] section of your `burpui.cfg` file to *true*:

```
[Global]
single = true
```

That's all you need to do for this mode to work.

## Multi-Agent

This mode allows you to access multiple [Burp](#) servers through the [bui-agent](#). The architecture is available on the [bui-agent](#) page.

To enable this mode, you need to set the *single* parameter of the [Global] section of your `burpui.cfg` file to *false*:

```
[Global]
single = false
```

Once this mode is enabled, you have to create **one** [Agent] section **per** agent you want to connect to in your `burpui.cfg` file:

```
# If you set single to 'false', add at least one section like this per
# bui-agent
[Agent:agent1]
# bui-agent address
host = 192.168.1.1
# bui-agent port
port = 10000
# bui-agent password
password = azerty
# enable SSL
ssl = true

[Agent:agent2]
# bui-agent address
host = 192.168.2.1
# bui-agent port
port = 10000
# bui-agent password
password = ytreza
# enable SSL
ssl = true
```

---

**Note:** The sections must be called [Agent : <label>] (case sensitive)

---

To configure your agents, please refer to the [bui-agent](#) page.

## 1.8.7 Versions

Burp-UI ships with two different backends:

- *Burpl*

- *Burp2*

These backends allow you to either connect to a [Burp](#) server version 1.x.x or 2.x.x.

---

**Note:** If you are using a [Burp](#) server version 2.x.x you **have** to use the *Burp2* backend, no matter what [Burp](#)'s protocol you are using.

---

### Burp1

---

**Note:** Make sure you have read and understood the [requirements](#) first.

---

The *burp-1* backend can be enabled by setting the *version* option to *1* in the [Global] section of your [burpui.cfg](#) file:

```
[Global]
version = 1
```

Now you can refer to the *Options* section for further setup.

### Burp2

---

**Note:** Make sure you have read and understood the [requirements](#) first.

---

---

**Note:** The [gunicorn](#) documentation may help you configuring your system.

---

The *burp-2* backend can be enabled by setting the *version* option to *2* in the [Global] section of your [burpui.cfg](#) file:

```
[Global]
version = 2
```

Now you can refer to the *Options* section for further setup.

### Options

```
# burp backend specific options
[Burp]
# burp status address (can only be '127.0.0.1' or ':::1')
bhost = :::1
# burp status port
bport = 4972
# burp binary
burpbin = /usr/sbin/burp
# vss_strip binary
stripbin = /usr/sbin/vss_strip
# burp client configuration file used for the restoration (Default: None)
bconfcli = /etc/burp/burp.conf
```

(continues on next page)

(continued from previous page)

```
# burp server configuration file used for the setting page
bconfsrv = /etc/burp/burp-server.conf
# temporary directory to use for restoration
tmpdir = /tmp
# how many time to wait for the monitor to answer (in seconds)
timeout = 5
```

Each option is commented, but here is a more detailed documentation:

- *bhost*: The address of the [Burp](#) server. In burp-1.x.x, it can only be *127.0.0.1* or *::1*
- *bport*: The port of [Burp](#)'s status port.
- *burpbin*: Path to the [Burp](#) binary (used for restorations).
- *stripbin*: Path to the [Burp](#) *vss\_strip* binary (used for restorations).
- *bconfcli*: Path to the [Burp](#) client configuration file (see [restoration](#)).
- *bconfsrv*: Path to the [Burp](#) server configuration file.
- *tmpdir*: Path to a temporary directory where to perform restorations.
- *timeout*: Time to wait for the monitor to answer in seconds.

## 1.8.8 Authentication

[Burp-UI](#) provides some authentication backends in order to restrict access only to granted users. There are currently three different backends:

- *LDAP*
- *Basic*
- *Local*

To disable the *authentication* backend, set the *auth* option of the [Global] section of your [burpui.cfg](#) file to *none*:

```
[Global]
auth = none
```

You can use multiple backends, they will be sorted by priority or in the order they are defined if no priority is found. If a user is present in several backends, the first one that matches both login and password will be used.

Example:

```
[Global]
auth = basic,ldap
```

### LDAP

The *ldap* authentication backend has some dependencies, please refer to the [requirements](#) page. To enable this backend, you need to set the *auth* option of the [Global] section of your [burpui.cfg](#) file to *ldap*:

```
[Global]
auth = ldap
```

Now you can add *ldap* specific options:

```
# ldapauth specific options
[LDAP]
# Backend priority. Higher is first
priority = 50
# LDAP host
host = 127.0.0.1
# LDAP port
port = 389
# Encryption type to LDAP server (none, ssl or tls)
# - try tls if unsure, otherwise ssl on port 636
encryption = tls
# specifies if the server certificate must be validated, values can be:
# - none (certificates are ignored)
# - optional (not required, but validated if provided)
# - required (required and validated)
validate = none
# SSL or TLS version to use, can be one of the following:
# - SSLv2
# - SSLv3
# - SSLv23
# - TLSv1
# - TLSv1_1 (Available only with openssl version 1.0.1+, requires python 2.7.9 or
↳higher)
version = TLSv1
# the file containing the certificates of the certification authorities
cafile = none
# Attribute to use when searching the LDAP repository
#searchattr = sAMAccountName
searchattr = uid
# LDAP filter to find users in the LDAP repository
# - {0} will be replaced by the search attribute
# - {1} will be replaced by the login name
filter = (&({0}={1})(burpui=1))
#filter = (&({0}={1})(|(userAccountControl=512)(userAccountControl=66048)))
# LDAP base
base = "ou=users,dc=example,dc=com"
# Binddn to list existing users
binddn = "cn=admin,dc=example,dc=com"
# Bindpw to list existing users
bindpw = Sup3rS3cr3tPa$$w0rd
```

---

**Note:** The *host* options accepts URI style (ex: `ldap://127.0.0.1:389`)

---

**Warning:** The quotes (“) around *base* and *binddn* are **MANDATORY**

## Basic

In order for the *basic* authentication backend to be enabled, you need to set the *auth* option of the [Global] section of your `burpui.cfg` file to *basic*:

```
[Global]
auth = basic
```



Now you can add *basic* specific options:

```
# basicauth specific options
# Note: in case you leave this section commented, the default login/password
# is admin/admin
[BASIC]
# Backend priority. Higher is first
priority = 100
admin = pbkdf2:sha1:1000$12345678$password
user1 = pbkdf2:sha1:1000$87654321$otherpassword
```

**Note:** Each line defines a new user with the *key* as the username and the *value* as the password

**Warning:** Since v0.3.0, passwords must be hashed (see [manage](#) to know how to create new users with hashed passwords)

## Local

In order for the *local* authentication backend to be enabled, you need to set the *auth* option of the [Global] section of your `burpui.cfg` file to *local*:

```
[Global]
auth = local
```

Now you can add *local* specific options:

```
# localauth specific options
# Note: if not running as root, then burp-ui must be run as group 'shadow' to
# allow PAM to work
[LOCAL]
# Backend priority. Higher is first
priority = 0
# List of local users allowed to login. If you don't set this setting, users
# with uid greater than limit will be able to login
users = user1,user2
# Minimum uid that will be allowed to login
limit = 1000
```

## 1.8.9 ACL

Burp-UI implements some mechanisms to restrict access on some resources only for some users. There is currently only one backend:

- *Basic ACL*

To disable the *acl* backend, set the *acl* option of the [Global] section of your `burpui.cfg` file to *none*:

```
[Global]
acl = none
```

The *ACL* engine has some settings as bellow:

```
# acl engine global options
[ACL]
# Enable extended matching rules (enabled by default)
# If the rule is a string like 'user1 = desk*', it will match any client that
# matches 'desk*' no mater what agent it is attached to.
# If it is a coma separated list of strings like 'user1 = desk*,laptop*' it
# will match the first matching rule no mater what agent it is attached to.
# If it is a dict like:
# user1 = '{"agents": ["srv*", "www*"], "clients": ["desk*", "laptop*"]}'
# It will also validate against the agent name.
extended = true
# If you don't explicitly specify ro/rw grants, what should we assume?
assume_rw = true
# The inheritance order matters, it means depending the order you choose,
# the ACL engine won't handle the grants the same way.
# By default, ACL inherited by groups will have lower priority, unless you
# choose otherwise
inverse_inheritance = false
# If you specify agents and clients separately, should we link them implicitly?
# For instance, '{"agents": ["agent1", "agent2"], "clients": ["client1", "client2"]}'
# will become: '{"agents": {"agent1": ["client1", "client2"], "agent2": ["client1",
→ "client2"]}}'
implicit_link = true
# Enable 'legacy' behavior
# Since v0.6.0, if you don't specify the agents name explicitly, users will be
# granted on every agents where a client matches user's ACL. If you enable the
# 'legacy' behavior, you will need to specify the agents explicitly.
# Note: enabling this option will also disable the extended mode
legacy = false
```

## Basic ACL

The *basic* acl backend can be enabled by setting the *acl* option of the `[Global]` section of your `burpui.cfg` file to *basic*:

```
[Global]
acl = basic
```

Now you can add *basic acl* specific options:

```
# basicacl specific options
# Note: in case you leave this section commented, the user 'admin' will have
# access to all clients whereas other users will only see the client that have
# the same name
[BASIC:ACL]
# Backend priority. Higher is first
priority = 100
# List of administrators
admin = user1,user2
# List of moderators. Users listed here will inherit the grants of the
# group '@moderator'
+moderator = user5,user6
@moderator = '{"agents":{"ro":["agent1"]}}'
# NOTE: if you are running single-agent mode, you should specify the ro/rw
# rights of the moderators using this special 'local' agent name:
# NOTE: this is the default when running single-agent mode if you don't
```

(continues on next page)

(continued from previous page)

```
# specify anything else
#@moderator = '{"agents": {"rw": "local"}}'
# Please note the double-quotes and single-quotes on the following lines are
# mandatory!
# You can also overwrite the default behavior by specifying which clients a
# user can access
# Suppose you are running single-agent mode (the default), you only need to
# specify a list of clients a user can access:
user3 = '{"clients": {"ro": ["prod*"], "rw": ["dev*", "test1"]}}'
# In case you are not in a single mode, you can also specify which clients
# a user can access on a specific Agent
user4 = '{"agents": {"agent1": ["client6", "client7"], "agent2": ["client8"]}}'
# You can define read-only and/or read-write grants using:
user5 = '{"agents": {"www*": {"ro": ["desk*"], "rw": ["desk1"]}}}'
# Finally, you can define groups using the syntax "@groupname" and adding
# members using "+groupname". Note: groups can inherit groups!
@group1 = '{"agents": {"ro": ["*"]}}'
@group2 = '{"clients": {"rw": ["dev*"]}}'
+group1 = @group2
+group2 = user5
# As a result, user5 will be granted the following rights:
# '{"ro": {"agents": ["*", "agent1"], "www*": ["desk*"]}, "rw": {"clients": ["dev*"],
↪ "www*": ["desk1"]}}'
```

**Warning:** The double-quotes and single-quotes are **MANDATORY**

By default, if a user is named `admin` it will be granted the admin role. Here are the default grants:

1. *admin* => you can do anything
2. *non admin* => you can only see the client that matches your username
3. *custom* => you can manually assign username to clients using the syntax `username = '{"agents": {"agent1": ["client1-1"], "agent2": ["client2-3", "client2-4"]}}'` (if you are running a multi-agent setup)
4. *moderators* => can edit the Burp server configurations of any agent unless told other wise (with `ro` rights), but cannot restore files unless told otherwise (with `rw` rights). Besides, moderators can create new users. They can also delete backups if they have `rw` rights on the client.

Since *v0.6.0*, you can define advanced grants through the `rw` and `ro` keyword.

- `ro` means you can only see backup stats and reports (this is great for monitoring teams/tools)
- `rw` means you can interact with the server in some way. For the *regular* users, `rw` means you can perform file restorations. For moderators, `rw` means you can delete backups (if burp thinks they are deletable), you can also create/update/delete client configuration files.

About the `inverse_inheritance` option, here is a concrete example. We assume you have this piece of configuration:

```
[ACL]
inverse_inheritance = false

[BASIC:ACL]
example = '{"agents": {"test": {"rw": ["demo"]}}}'
```

(continues on next page)

(continued from previous page)

```
@gp_ro = '{"agents": {"*": {"ro": ["*"]}}}'
+gp_ro = example
```

Then the client demo on the test agent will be granted rw rights, anything else will be ro. Now if you set `inverse_inheritance = true`, the @gp\_ro grants will have the highest priority, meaning the client demo on the test agent will be granted ro rights like any other client.

Please also note the order of your rules matters (although the UI is unable to re-order your rules). For instance, this:

```
[BASIC:ACL]
user1 =
@gp1 = '{"clients": {"rw": ["tata", "titi"]}}'
+gp1 = user1
@gp2 = '{"clients": {"ro": ["*"]}, "agents": {"rw": "local"}}'
+gp2 = @gp1
```

Is not the same as:

```
[BASIC:ACL]
user1 =
@gp2 = '{"clients": {"ro": ["*"]}, "agents": {"rw": "local"}}'
+gp2 = @gp1
@gp1 = '{"clients": {"rw": ["tata", "titi"]}}'
+gp1 = user1
```

## 1.9 Manage

Since *v0.3.0*, Burp-UI ships with a tool called `bui-manage`. This tool allows you to create new users and to manage database migrations.

This tool is actually a wrapper script that interacts with the core of Burp-UI. You can use it like this:

```
bui-manage [wrapper options...] [--] <subcommand>
```

This page details the *subcommand* usage. The tool provides some inline help too:

```
bui-manage -h
usage: bui-manage [-h] [-c <CONFIG>] [-i <MIGRATIONS_DIR>]
                  [-m <agent|server|worker|manage>]
                  ...

positional arguments:
  remaining

optional arguments:
  -h, --help            show this help message and exit
  -c <CONFIG>, --config <CONFIG>
                        burp-ui configuration file
  -i <MIGRATIONS_DIR>, --migrations <MIGRATIONS_DIR>
                        migrations directory
  -m <agent|server|worker|manage>, --mode <agent|server|worker|manage>
                        application mode
```

```
# note the -- used to separate the wrapper from the actual command
bui-manage -- --help
Usage: flask [OPTIONS] COMMAND [ARGS]...

This shell command acts as general utility script for Flask applications.

It loads the application configured (either through the FLASK_APP
environment variable) and then provides commands either provided by the
application or Flask itself.

The most useful commands are the "run" and "shell" command.

Example usage:

$ export FLASK_APP=hello
$ export FLASK_DEBUG=1
$ flask run
```

Options:

```
--help  Show this message and exit.
```

Commands:

compile_translation	Compile translations.
create_user	Create a new user.
db	Perform database migrations.
diag	Check Burp-UI is correctly setup.
hash_password	Hash a given password to fill the...
init_translation	Initialize a new translation for the given...
legacy	Legacy server for backward compatibility.
run	Runs a local development server for the...
setup_burp	Setup burp client for burp-ui.
shell	Runs a shell in the app context.
sysinfo	Returns a couple of system informations to...
update_translation	Update translation files.
websocket	Start a new websocket server.

### 1.9.1 Database

To manage database migration, you first need to enable database support within your configuration file (see [Production section](#))

You will also need some extra requirements:

```
pip install --upgrade "burp-ui[sql]"
```

Then you just have to run the following command to have your database setup:

```
bui-manage db upgrade
```

If your configuration is not in a *common* location, you can specify it like this:

```
bui-manage -c path/to/burpui.cfg db upgrade
```

If you did not install [Burp-UI](#) in a *common* location or you want to run it without installing it directly through the sources, you may need to specify the location of the *migrations* scripts like this:

```
bui-manage -c path/to/burpui.cfg -i path/to/migrations db upgrade
```

## 1.9.2 Users

You can create new users using the `bui-manage` file like this:

```
bui-manage create_user <new_username>
```

By default, the script will create new users for the **Basic** authentication backend. Without further details, a new password will be generated. You can either provide a password through the command line or tell the script to ask you what to setup using either the `-p` or `-a` options.

Examples:

```
bui-manage create_user user1
[*] Adding 'user1' user...
[+] Generated password: 7lVIanuJ
[+] Success: True

bui-manage create_user -p toto user2
[*] Adding 'user2' user...
[+] Success: True

bui-manage create_user -a user3
[*] Adding 'user3' user...
Password:
Confirm:
[+] Success: True
```

## 1.9.3 Hashing passwords

Since *v0.3.0*, the **BASIC** authentication backend supports hashed passwords. Support for *legacy* plain-text passwords **will be removed in \*v0.7.0\***.

In order to automate user creation using external scripts, a new `bui-manage` subcommand has been introduced in *v0.6.0*.

Usage:

```
Usage: flask hash_password [OPTIONS]

    Hash a given password to fill the configuration file.

Options:
  -p, --password TEXT  Password to assign to user.
  -u, --username TEXT  Provide the username to get the full configuration
                        line.
  -b, --batch           Don't be extra verbose so that you can use the output
                        directly in your scripts. Requires both -u
➔ and -p.
  --help               Show this message and exit.
```

Examples:

```
$ bui-manage hash_password --password demo --username demo --batch
demo = pbkdf2:sha256:50000$w5jD2WT1
↪$d89cd4da1b6c3c3f05173faaf3feea802eee7b5b9a378ae8987d5ac6676d166b

$ bui-manage hash_password --password demo --username demo
'demo' hashed into: pbkdf2:sha256:50000$feQoDIIP
↪$d42e056b75092beda52736ce792c6518061b21d6e1b2c755fdd71a764184ebd9
#8<-----
demo = pbkdf2:sha256:50000$feQoDIIP
↪$d42e056b75092beda52736ce792c6518061b21d6e1b2c755fdd71a764184ebd9
#8<-----

$ bui-manage hash_password -u demo
Password:
'*****' hashed into: pbkdf2:sha256:50000$MNH0gtldr
↪$22d64b17bb8135be3ad61249d649e89ed96a5d4878b23e4889be24dfc062c126
#8<-----
demo = pbkdf2:sha256:50000$MNH0gtldr
↪$22d64b17bb8135be3ad61249d649e89ed96a5d4878b23e4889be24dfc062c126
#8<-----

$ bui-manage hash_password
Password:
'*****' hashed into: pbkdf2:sha256:50000$eWzSCvSW
↪$9df9cdf854461a5f927708475e0470ad648f8cd0613ca61f8cee639db165ab53
```

## 1.9.4 Configure

Since *v0.4.0*, the `bui-manage` tool is now able to help you setup both [Burp](#) and [Burp-UI](#) so they speak to each other.

The available options are:

```
bui-manage setup_burp --help

Usage: flask setup_burp [OPTIONS]

    Setup burp client for burp-ui.

Options:
  -b, --burp-conf-cli TEXT    Burp client configuration file
  -s, --burp-conf-serv TEXT   Burp server configuration file
  -c, --client TEXT           Name of the burp client that will be used by
                              Burp-UI (defaults to "bui")
  -h, --host TEXT             Address of the status server (defaults to "":1")
  -r, --redis TEXT            Redis URL to connect to
  -d, --database TEXT         Database to connect to for persistent storage
  -p, --plugins TEXT          Plugins location
  -n, --dry                   Dry mode. Do not edit the files but display
                              changes
  --help                      Show this message and exit.
```

The script needs the [Burp](#) configuration files to be readable **AND** writable.

---

**Note:** This script was initially developped to setup the docker image. I do not guarantee to be able to support it out of the docker context.

---

---

**Note:** This script only supports Burp 2.0.x.

---

The docker image uses this script like this:

```
bui-manage -c $BURPUI_CONFIG setup_burp -b $BURP_CLIENT_CONFIG \  
-s $BURP_SERVER_CONFIG -h $BURP_SERVER_ADDR -c $BURPUI_CLIENT_NAME \  
-r $REDIS_SERVER -d $DATABASE_URL
```

### 1.9.5 WebSocket

---

**Note:** This tool first appeared with [Burp-UI v0.6.0](#).

---

Launch a dedicated websocket server so you can have more than one worker. You may need a dedicated reverse-proxy though.

Example:

```
bui-manage websocket --help  
Usage: flask websocket [OPTIONS]  
  
    Start a new websocket server.  
  
Options:  
  -b, --bind TEXT      Which address to bind to for the websocket server  
  -p, --port INTEGER   Which port to listen on for the websocket server  
  -d, --debug          Whether to start the websocket server in debug mode  
  --help              Show this message and exit.
```

### 1.9.6 Sysinfo

---

**Note:** This tool first appeared with [Burp-UI v0.5.0](#).

---

This tool will help you to gather system informations in order to make a detailed bug report.

Example:

```
bui-manage sysinfo  
  
Python version: 2.7.9  
Burp-UI version: 0.6.0 (stable)  
Single mode: True  
Backend version: 2  
Config file: share/burpui/etc/burpui.sample.cfg
```

You can also add the `-v` flag while running `sysinfo` but please **MAKE SURE NO SENSITIVE DATA GET EXPOSED**.

Example:



```

bui-manage sysinfo -v
Python version: 2.7.9
Burp-UI version: 0.6.0 (stable)
Single mode: True
Backend version: 2
Config file: share/burpui/etc/burpui.sample.cfg
>>>> Extra verbose informations:
!!! PLEASE MAKE SURE NO SENSITIVE DATA GET EXPOSED !!!

[Burp] section:
8<-----BEGIN
8<-----END

[Production] section:
8<-----BEGIN
storage = default
session = default
cache = default
redis = localhost:6379
celery = false
database = none
limiter = false
ratio = 60/minute
8<-----END

[Global] section:
8<-----BEGIN
version = 2
single = true
auth = basic
acl = basic
prefix = none
plugins = none
8<-----END

```

### 1.9.7 Diag

---

**Note:** This tool first appeared with [Burp-UI v0.5.0](#).

---

This tool will help you detect misconfiguration. It will **not** modify your files, you will have to use the [Configure](#) tool for that.

The available options are:

```

bui-manage diag --help

Usage: flask diag [OPTIONS]

    Check Burp-UI is correctly setup

Options:
  -c, --client TEXT  Name of the burp client that will be used by Burp-UI
                     (defaults to "bui")
  -h, --host TEXT    Address of the status server (defaults to "::<1")

```

(continues on next page)

(continued from previous page)

```
-t, --tips      Show you some tips
--help         Show this message and exit.
```

### Examples:

```
bui-manage diag

The cname of your burp client does not match: hydrogen != bui
The burp server address does not match: 127.0.0.1 != ::1
'max_status_children' is to low, you need to set it to 15 or more. Please edit your /
↳etc/burp/burp-server.conf file
Your burp client is not listed as a 'restore_client'. You won't be able to view other_
↳clients stats!
For performance reasons, it is recommended to enable the 'monitor_browse_cache'
Unable to find the /etc/burp/clientconfdir/bui file
Some errors have been found in your configuration. Please make sure you ran this_
↳command with the right flags! (see --help for details)

bui-manage diag -c hydrogen -h 127.0.0.1 -t

'max_status_children' is to low, you need to set it to 15 or more. Please edit your /
↳etc/burp/burp-server.conf file
Your burp client is not listed as a 'restore_client'. You won't be able to view other_
↳clients stats!
For performance reasons, it is recommended to enable the 'monitor_browse_cache'
Well, if you are sure about your settings, you can run the following command to help_
↳you setup your Burp-UI agent. (Note, the '--dry' flag is here to show you the_
↳modifications that will be applied. Once you are OK with those, you can re-run the_
↳command without the '--dry' flag):
    > bui-manage setup_burp --host="127.0.0.1" --client="hydrogen" --dry
```

When your configuration is OK, you should see this message:

```
Congratulations! It seems everything is alright. Burp-UI should run without any issue_
↳now.
```

## 1.10 Celery

Since v0.3.0, **Burp-UI** supports asynchronous operations thanks to **Celery**. In order to use this feature, you need to enable it in the configuration (see [Production](#) section)

You will also need some extra requirements:

```
pip install "burp-ui[celery]"
```

**Celery** needs a *Broker* to communicate between the workers and your application. I chose **Redis** so you will need a working **Redis** server (Basically you just need to run `apt-get install redis-server` on Debian based distributions)

### 1.10.1 Runner

Once everything is setup, you need to launch a worker. **Burp-UI** ships with a helper script called `bui-celery`. You can use it like this:

```
bui-celery --beat
```

If your configuration is not in a *common* location, you can specify it like this:

```
bui-celery -c path/to/burpui.cfg -- --beat
```

---

**Note:** A systemd service example file is shipped in the *contrib* directory

---

---

**Note:** The `--beat` option is recommended since some operations need to be executed periodically

---

---

**Note:** The usage of a [database](#) is recommended to keep a track of executed tasks

---

## 1.11 WebSocket

Since *v0.6.0*, [Burp-UI](#) supports WebSockets for better/smarter notifications. In order to use this feature, you need some extra requirements:

```
pip install "burp-ui[websocket]"
```

It is highly recommended to use a [Redis Broker](#) for the websocket server to interact with the celery workers and other pieces of the code. It is also advised to run one or several dedicated WebSocket servers behind a reverse-proxy because *gunicorn* does not play well with it.

The details of the configuration may be found in the [WebSocket](#) section.

### 1.11.1 Dedicated Server

You can choose to either run an embedded WebSocket server though this is not recommended in production or you can run one or several dedicated WebSocket servers through the `bui-manage` command like this:

```
bui-manage websocket --bind 0.0.0.0 --port 5001
```

If you are running the above command, you'll need to set the `url` option under the `[WebSocket]` section to `"document.domain + ':5001'"` (unless you use a reverse-proxy, see below).

**Warning:** The quotes are **MANDATORY** in this case.

Alternatively, you can setup a reverse-proxy as explained below.

---

**Note:** A systemd service example file is shipped in the *contrib* directory

---

### 1.11.2 Reverse-proxy

Running one or several dedicated WebSocket server is the recommended setup in production. You will find more details on this in the [Flask-SocketIO](#) documentation.

Running a dedicated server on a dedicated port and/or IP may be a pain. That's the reason why you can/should setup a reverse-proxy in front of this using a configuration like:

```
server {
    listen 80;
    server_name _;

    location / {
        proxy_pass http://127.0.0.1:5000;
    }

    location /socket.io {
        proxy_http_version 1.1;
        proxy_buffering off;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_pass http://127.0.0.1:5001/socket.io;
    }
}
```

If you ran several servers, you can use this config:

```
upstream socketio_nodes {
    ip_hash;

    server 127.0.0.1:5001;
    server 127.0.0.1:5002;
    # to scale the app, just add more nodes here!
}

server {
    listen 80;
    server_name _;

    location / {
        proxy_pass http://127.0.0.1:5000;
    }

    location /socket.io {
        proxy_http_version 1.1;
        proxy_buffering off;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_pass http://socketio_nodes/socket.io;
    }
}
```

## 1.12 Gunicorn

Starting from *v0.0.6*, [Burp-UI](#) supports [Gunicorn](#) in order to handle multiple users simultaneously because some operations (like the online restoration) may take some time and thus may block any further requests. With [Gunicorn](#),

you have several workers that can proceed the requests so you can handle more users.

You need to install `gunicorn` and `gevent`:

```
pip install "burp-ui[gunicorn]"
```

You will then be able to launch **Burp-UI** this way:

```
gunicorn -k gevent -w 4 'burpui:create_app(conf="/path/to/burpui.cfg")'
```

**Note:** If you decide to use `gunicorn` AND the embedded websocket server, you'll need to use the `geventwebsocket.gunicorn.workers.GeventWebSocketWorker` worker name (ie. `-k geventwebsocket.gunicorn.workers.GeventWebSocketWorker`). For this worker to be available, you need to install the `gevent-websocket` `pip` package.

This will listen to `127.0.0.1:8000` by default. See the `gunicorn` [documentation](#) for details.

You will need to add the `-b 0.0.0.0:5000` option in order to listen on all interfaces on port 5000 for instance.

When using `gunicorn`, the command line options are not available. Instead, you run the **Burp-UI** `create_app` method directly. Here are the parameters you can play with:

- `conf`: Path to the **Burp-UI** configuration file
- `verbose`: Verbosity level between 0 and 4
- `logfile`: Path to a logfile in order to log **Burp-UI** internal messages
- `reverse_proxy`: Whether we are behind a reverse-proxy or not (defaults to `True`)

**Warning:** You **MUST** set the `appsecret` option in your configuration file when using `gunicorn`. The default *magic* value 'random' cannot be used. If you don't change the settings the default value will be 'none' and your cookies won't be secured.

### 1.12.1 Advanced usage

`Gunicorn` supports further settings (see its [documentation](#) for details). For instance, you would probably like to use the `-c` flag with the sample configuration file bundled with **Burp-UI** in `contrib/gunicorn/burpui_gunicorn.py`.

Usage example:

```
gunicorn -c burpui_gunicorn.py 'burpui:create_app(conf="/etc/burp/burpui.cfg", logfile=
↳ "/var/log/gunicorn/burp-ui_info.log")'
```

### 1.12.2 Daemon

If you wish to run **Burp-UI** as a daemon process, the recommended way is to use `Gunicorn`.

#### Requirements

First of all, you'll need a dedicated user.

```
# create the burpui user
useradd -m -r -d /var/lib/burpui -c 'Burp-UI daemon user' burpui
mkdir /etc/burp
# copy the burp-ui sample configuration file
cp /usr/local/share/burpui/etc/burpui.sample.cfg /etc/burp/burpui.cfg
mkdir -p /var/log/gunicorn
chown -R burpui: /var/log/gunicorn
```

You will also need a custom client configuration and you will have to create the certificates accordingly:

```
# create the configuration file used by burp-ui
cat >/var/lib/burpui/burp.conf<<EOF
mode = client
port = 4971
status_port = 4972
server = 127.0.0.1
password = abcdefgh
cname = bui-agent1
pidfile = /var/lib/burpui/bui-agent1.client.pid
syslog = 0
stdout = 1
progress_counter = 1
ca_burp_ca = /usr/sbin/burp_ca
ca_csr_dir = /var/lib/burpui/CA-client
ssl_cert_ca = /var/lib/burpui/ssl_cert_ca-client.pem
ssl_cert = /var/lib/burpui/ssl_cert-client.pem
ssl_key = /var/lib/burpui/ssl_cert-client.key
ssl_peer_cn = burpserver
EOF
# generate the certificates
burp_ca --name bui-agent1 --ca burpCA --key --request --sign --batch
cp /etc/burp/ssl_cert_ca.pem /var/lib/burpui/ssl_cert_ca-client.pem
cp -a /etc/burp/CA/bui-agent1.crt /var/lib/burpui/ssl_cert-client.pem
cp -a /etc/burp/CA/bui-agent1.key /var/lib/burpui/ssl_cert-client.key
chown -R burpui: /var/lib/burpui/
```

Now you need to add the *bui-agent1* client to the authorized clients:

```
echo "password = abcdefgh" >/etc/burp/clientconfdir/bui-agent1
echo "restore_client = bui-agent1" >>/etc/burp/burp-server.conf
```

You will also need to increase the number of status clients by setting `max_status_children` to 15:

```
echo "max_status_children = 15" >>/etc/burp/burp-server.conf
```

Finally, make sure you set `bconfcli: /var/lib/burpui/burp.conf` in your **Burp-UI** configuration file (*/etc/burp/burpui.cfg*).

If you want to take advantage of *advanced* features such as client add/removal and configuration files edition, you should set the permissions accordingly Burp-side.

First of all, add the following lines in your */etc/burp/burp-server.conf*:

```
user = burpui
group = burpui
```

Then you need to fix some permissions:

```
chown -R burpui: /etc/burp/{burp-server.conf,burpui.cfg,CA,CA.cnf,clientconfdir,
↪dhfile.pem,ssl_cert_ca.pem,ssl_cert-server.key,ssl_cert-server.pem} /var/spool/burp
chgrp burpui /etc/burp
chmod g+rxw /etc/burp
```

Finally you can restart your burp-server.

---

**Note:** The above commands are meant for *default* setup. You may need to adapt the paths.

---

## Debian-style

When installing the *gunicorn* package on debian, there is a handler script that is able to start several instances of *Gunicorn* as daemons.

All you need to do is installing the *gunicorn* package and adding a configuration file in */etc/gunicorn.d/*.

There is a sample configuration file available [here](#).

```
# install the gunicorn package
apt-get install gunicorn
# copy the gunicorn sample configuration
cp /usr/local/share/burpui/contrib/gunicorn.d/burp-ui /etc/gunicorn.d/
# now restart gunicorn
service gunicorn restart
```

## Systemd

If you are not running on debian or you prefer not to use the gunicorn debian package, the handler script may not be available. You will then have to create your own service. We can do this for systemd for example:

```
# copy the gunicorn configuration file
cp /usr/local/share/contrib/gunicorn/burpui_gunicorn.py /etc/burp/
# create the service file
cat >/etc/systemd/service/bui-gunicorn.service<<EOF
[Unit]
Description=Burp-UI gunicorn service
After=network.target

[Service]
User=burpui
Group=burpui
ExecStart=/usr/local/bin/gunicorn -c /etc/burp/burpui_gunicorn.py 'burpui:create_
↪app(conf="/etc/burp/burpui.cfg",logfile="/var/log/gunicorn/burp-ui_info.log") '

[Install]
WantedBy=multi-user.target
EOF
# enable the new service
systemctl enable bui-gunicorn.service
# start the service
systemctl start bui-gunicorn.service
```

### 1.12.3 Reverse-Proxy

You may want to add a reverse-proxy so **Burp-UI** can be accessed on port 80 (or 443) along with other applications.

Here is a sample configuration for Nginx:

```
server {
    listen 80;
    server_name burpui.example.com;

    access_log /var/log/nginx/burpui.access.log;
    error_log /var/log/nginx/burpui.error.log;

    location / {

        # you need to change this to "https", if you set "ssl" directive to "on"
        proxy_set_header    X-FORWARDED_PROTO http;
        proxy_set_header    Host                $http_host;
        proxy_set_header    X-Forwarded-For    $remote_addr;

        proxy_read_timeout 300;
        proxy_connect_timeout 300;

        proxy_pass http://localhost:5000;
    }
}
```

#### Sub-root path

You can host **Burp-UI** behind a sub-root path. For instance `/burpui`. To accomplish this, you can either setup your reverse-proxy to announce the desired *prefix*, or you can use the `prefix` option in your **Burp-UI** configuration file (see [usage](#) for details).

If you want to configure this reverse-proxy side, you need to announce the HTTP Header `X-Script-Name`.

Here is a sample configuration for Nginx:

```
server {
    listen 80;
    server_name example.com;

    access_log /var/log/nginx/burpui.access.log;
    error_log /var/log/nginx/burpui.error.log;

    location /burpui {

        # you need to change this to "https", if you set "ssl" directive to "on"
        proxy_set_header    X-FORWARDED_PROTO http;
        proxy_set_header    Host                $http_host;
        proxy_set_header    X-Forwarded-For    $remote_addr;
        # Our service is hosted behind the "/burpui" prefix
        proxy_set_header    X-Script-Name      /burpui;

        proxy_read_timeout 300;
        proxy_connect_timeout 300;

        proxy_pass http://localhost:5000;
    }
}
```

(continues on next page)



(continued from previous page)

```
}
}
```

Apache sample:

```
ProxyPass /burp/ http://localhost:5000/burp/
ProxyPassReverse /burp/ http://localhost:5000/burp/
<Location /burp/>
    SetOutputFilter proxy-html
    ProxyPassReverse /burp/
    ProxyHTMLURLMap http://localhost:5000/      /
    Require all granted
</Location>
```

**Warning:** If your *prefix* does not start with a '/', it will be ignored.

## 1.12.4 Production

We can consider the [demo](#) as a production example of what you can setup/expect in your environment. It is using [Gunicorn](#) along with [Nginx](#) as described above.

In order to improve performances, [Redis](#) can be used to cache sessions and various API calls. You can also enable the [celery](#) worker for asynchronous jobs. Additionally, you can enable the [SQL](#) storage.

The [FAQ](#) answers these questions:

- [Why using redis?](#)
- [Why using SQL?](#)
- [Why using Celery?](#)

See the [production](#) section of the [usage](#) page.

## 1.13 Docker

Since the *v0.4.0*, a docker image is provided. It ships with the latest stable release of [Burp-UI](#) and supports the celery worker introduced in *v0.3.0* if you link it to a redis container.

### 1.13.1 Introduction

All you need is [docker](#) and [docker-compose](#). A *docker-compose.yml* file is provided. There are a few variables supported to setup your system:

- **BURPUI\_CONFIG** - Specify where the [Burp-UI](#) configuration file is located. It defaults to `"/etc/burp/burpui.cfg"`.
- **BURPUI\_VERBOSE** - Specify the log verbosity (between 0 and 4). It defaults to 0.
- **BURPUI\_CLIENT\_NAME** - Specify the name of the burp client that will be used by [Burp-UI](#). It defaults to `"bui"`.
- **BURPUI\_UID** - uid of the *burpui* user you want to map in your host. It defaults to 5337.

- **BURPUI\_GID** - gid of the *burpui* group you want to map in your host. It defaults to 5337.
- **BURPUI\_PLUGINS** - Directory where to look for plugins. It defaults to none which means no plugins will be loaded.
- **BURPUI\_WS\_WORKERS** - How many WebSocket servers to spawn. Defaults to the number of CPU cores/sockets/threads found.
- **BURPUI\_RP\_SCHEME** - HTTP Scheme to set for the reverse-proxy. If you are behind a reverse-proxy that provides SSL, you should set this to *https* which is the default value.
- **BURP\_CLIENT\_CONFIG** - Specify the path of the burp client configuration file to use for the **Burp-UI** client. It defaults to “*/tmp/burp.conf*”. It means you won’t have access to it outside of the container. It is intended to not override the */etc/burp/burp.conf* file if you already use it.
- **BURP\_SERVER\_CONFIG** - Specify the path of the burp-server configuration file. It defaults to “*/etc/burp/burp-server.conf*”.
- **DATABASE\_URL** - Specify the URL of the database to connect to. It defaults to “*sqlite:///var/lib/burpui/store.db*”.
- **REDIS\_SERVER** - Specify the address of the redis server. It defaults to “*redis:6379*”.
- **BURP\_SERVER\_ADDR** - Specify the address of the burp-server status port. If set to “*auto*” (the default), we will use the address of the docker host. Make sure your status port is listening on this interface.
- **TIMEZONE** - Specify the timezone of your burp-server. It defaults to Europe/Paris.

The provided *docker-compose.yml* file suggests that you *mount* the */etc/burp* and */var/spool/burp* paths inside the container (this is automatic) so that **Burp-UI** is able to access some required files.

### 1.13.2 Requirements

The docker image only works with a burp server version 2.0.x.

*Burp-UI* will be launched with the user *burpui* inside the container. This user has the UID `$BURPUI_UID` and the GID `$BURPUI_GID` so you may want to create a user with the associated UID/GID in your host and make sure it has read **AND** write access to */etc/burp* and */var/spool/burp*.

For instance:

```
groupadd -g 5337 burpui
useradd -r -m -d /var/lib/burpui -c 'Burp-UI daemon user' -u 5337 -g 5337 burpui
```

### 1.13.3 Usage

All you have to do is to retrieve the *docker-compose.yml* file, edit the variables if needed and launch the docker containers. For instance you could do:

```
mkdir -p ~/workspace
cd ~/workspace
git clone https://git.ziirish.me/ziirish/burp-ui.git
cd burp-ui
docker-compose up -d
```

That’s it. Really. Now profit and go to <http://localhost:5000/>

### 1.13.4 Troubleshooting

Here are some hints to help you troubleshoot your **Burp-UI** container.

#### Cannot launch burp process: Unable to spawn burp process

This means **Burp-UI** was not able to spawn a burp client that is able to communicate with the server. You can check the containers logs using the `docker-compose logs` command. If the output contains something like:

```
It looks like your burp server is not exposing it's status port in a way that is
↳ reachable by Burp-UI!
You may want to set the 'status_address' setting with either '1.2.3.4', '::' or '0.0.
↳ 0.0' in the /etc/burp/burp-server.conf file in order to make Burp-UI work
```

It means your burp-server is not exposing its status port. The above output gives you the instructions to fix it.

**Note:** You'll have to restart your burp-server to bind to the new *status\_address*

Other errors may be reported as well by the `docker-compose logs` command. Please read its output carefully.

If the error still occurs, you may need to investigate further. You can run these commands:

```
# docker-compose ps
Name                        Command                  State      Ports
-----
burpui_burpui_1            /app/init app:start      Up         127.0.0.1:5000->5000/tcp
burpui_redis_1             docker-entrypoint.sh redis ... Up         6379/tcp
# docker exec -it burpui_burpui_1 /bin/bash
root@59d883806fc7:/# su - burpui
$ /usr/sbin/burp -c /tmp/burp.conf -a m
{ "logline": "Could not find ssl_cert_ca /etc/burp/ssl_cert_ca-client-bui.pem: No
↳ such file or directory" }
{ "logline": "auth ok" }
{ "logline": "Server version: 2.0.52" }
{ "logline": "nocsr ok" }
{ "logline": "SSL is using cipher: DHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=DH
↳ Au=RSA Enc=AESECM(256) Mac=AEAD" }
{ "logline": "Certificate doesn't verify." }
{ "logline": "check cert failed" }
```

In this case, we see a pem file is missing. You can usually fix this issue by running:

```
# ln -s /etc/burp/ssl_cert_ca.pem /etc/burp/ssl_cert_ca-client-bui.pem
```

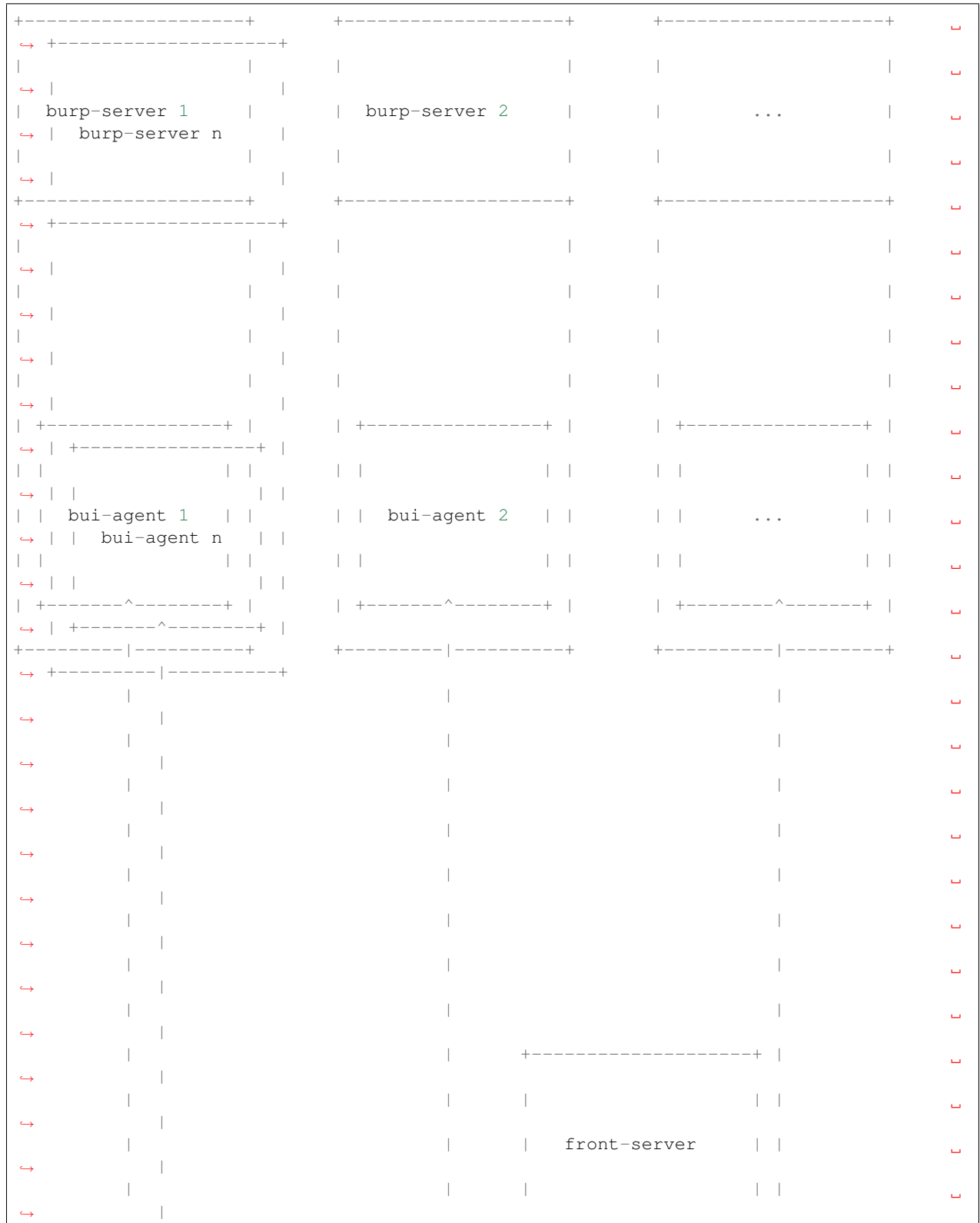
## 1.14 bui-agent

The **bui-agent** is a kind of proxy between a **Burp** server and your **Burp-UI** server.

It is useful when you have several servers to monitor and/or when you don't want (or can't) install the full **Burp-UI** on your server.

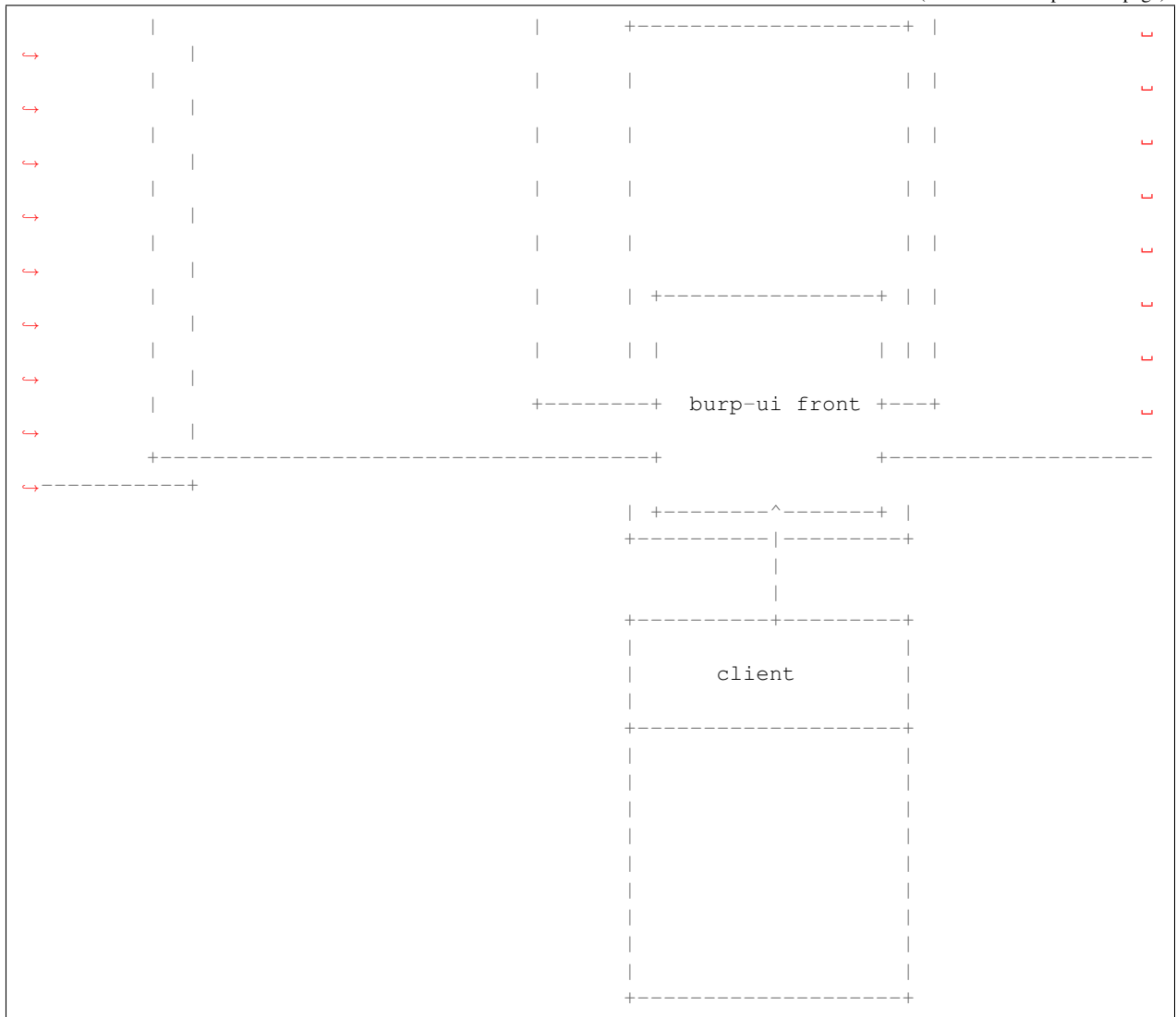
### 1.14.1 Architecture

The architecture is described bellow:



(continues on next page)

(continued from previous page)



### 1.14.2 Requirements

The agent is powered by gevent. In order to install it, you can run the following command:

```
pip install burp-ui-agent
```

**Note:** Starting with *v0.5.0*, the agent now has its own package.

### 1.14.3 Configuration

These agents must be launched on every server hosting a [Burp](#) instance you'd like to monitor.

They have a specific [buiagent.cfg](#) configuration file with a `[Global]` section as below:

```
[Global]
# On which port is the application listening
port = 10000
# On which address is the application listening
# ':' is the default for all IPv6
# set it to '0.0.0.0' if you want to listen on all IPv4 addresses
bind = ::
# enable SSL
ssl = true
# ssl cert
sslcert = /etc/burp/ssl_cert-server.pem
# ssl key
sslkey = /etc/burp/ssl_cert-server.key
# burp server version 1 or 2
version = 1
# agent password
password = password
```

Each option is commented, but here is a more detailed documentation:

- *port*: On which port is **bui-agent** listening.
- *bind*: On which address is **bui-agent** listening.
- *ssl*: Whether to communicate with the **Burp-UI** server over SSL or not.
- *sslcert*: What SSL certificate to use when SSL is enabled.
- *sslkey*: What SSL key to use when SSL is enabled.
- *version*: What version of **Burp** this **bui-agent** instance manages. (see **Burp-UI versions** for more details)
- *password*: The shared secret between the **Burp-UI** server and **bui-agent**.

As with **Burp-UI**, you need a specific section depending on the *version* value. Please refer to the **Burp-UI versions** section for more details.

### 1.14.4 Example

Here is a full usage example:

```
# On the server called 'agent1'
agent1:~$ bui-agent -c path/to/buiagent.cfg

# On the server called 'agent2'
agent2:~$ bui-agent -c path/to/buiagent.cfg

# On the server called 'front'
front:~$ burp-ui -c path/to/burpui.cfg
```

This example uses three servers. You then only need to point your browser to <http://front:5000/> for instance, and the **Burp-UI** instance (front) will *proxy* the requests to the two agents for you.

## 1.15 Service

I have no plan to implement daemon features, but there are a lot of tools available to help you achieve such a behavior. To run **bui-agent** as a service, a **systemd** file is provided. You can use it like this:

```
cp /usr/local/share/burpui/contrib/systemd/bui-agent.service /etc/systemd/system/
systemctl daemon-reload
systemctl enable bui-agent.service
systemctl start bui-agent.service
```

## 1.16 Contributing

Contributions are welcome. You can help in any way you want, for instance by opening issues on the [bug tracker](#), sending patches, etc.

There is also a dedicated website. Currently it only hosts a [Discourse](#) instance where you can discuss with each other. No need to create another account, the one you use in the [bug tracker](#) can be imported automatically!

Feel free to use it and post your tips and remarks.

The address is: <https://forum.burp-ui.org/>

You can financially support the project if you find it useful or if you would like to sponsor a feature. Details on my [website](#).

### 1.16.1 Translating

Translations are very welcome! If you are willing to help, you will need some tools:

```
pip install Flask-Babel
```

Then you need to fork the project retrieve the sources:

```
git clone https://git.ziirish.me/<your_login>/burp-ui.git
cd burp-ui
git submodule update --init
pip install -r requirements.txt
```

You can have the list of available languages by running:

```
ls burpui/translations
```

### New language

If your language is not listed, you can create a new translation running the following command:

```
./tools/bui-manage init_translation <country_code> # where <country_code> can be "de",
↪ "ru", etc.
```

### Update translation

If you want to update an existing (and/or un-complete) translation, you probably want to have a look at the *templates* files.

An un-translated file will contain things like:

```
<h1>Some title</h1>
```

The string *Some title* won't be translated as is. You need to update the template like this:

```
<h1>{{ _('Some title') }}</h1>
```

Then you can update the translation files with the following command:

```
./tools/bui-manage update_translation
```

### Translation

Now you can proceed the translation in the file *burpui/translations/<country\_code>/LC\_MESSAGES/messages.po*.

It looks like:

```
#: burpui/forms.py:18 burpui/templates/login.html:8
msgid "Username"
msgstr ""
```

You just have to put the translations in the *msgstr* line like:

```
#: burpui/forms.py:18 burpui/templates/login.html:8
msgid "Username"
msgstr "Utilisateur"
```

Once it's done, you can push the sources and create a Merge Request on GitLab:

```
git checkout -b translation-<country_code>
git add burpui/translations/<country_code>/LC_MESSAGES/messages.po
git commit -m "<country_code> translation"
git push -u origin translation-<country_code>
```

### 1.16.2 Issues / Bugs

If you find any issue while using Burp-UI please report it on the [bug tracker](#). All issues should contain the used command line to reproduce the problem, the debug output and both versions of burp and Burp-UI you are using.

A Bug Report template is available directly in the Issue Tracker:



## New Issue

The screenshot shows the 'New Issue' form. The 'Title' field has a dropdown menu with the text 'Choose a template'. The 'Description' field is open, displaying a search bar with the text 'Filter' and a magnifying glass icon. Below the search bar, there are four options: 'Bug' (which is highlighted), 'Feature Request', 'No template', and 'Reset template'.

The template illustrate the commands to run in order to provide as much details as you can in order to help reproduce the issue.

If you are running Burp-UI below *v0.5.0*, you can get those informations using the following commands:

```
$ /usr/sbin/burp -v
burp-1.4.40
$ burp-ui -V -v
burp-ui: v0.1.0.dev (90deb82c7b0be35f1a70bb073c9926b5947c6a85)
$ burp-ui -v
```

Optionally your python version and your OS might be useful as well.

### 1.16.3 Questions

Ask questions in the [discussion forum](#). Do not use the issue tracker for this purpose.

Burp-UI has extensive online documentation please read the [doc](#).

### 1.16.4 Troubleshooting

In case you encounter troubles with Burp-UI, you should run it with the `-vvvv` flag and paste the relevant output within your bug-report. Please also give the version of burp **AND** Burp-UI. Since *v0.0.6* you can use the `-V` or `--version` flag in order to get your version number.

## 1.16.5 Merge / Pull requests

I would like you to use [gitlab](#) for your Merge requests in order to take advantage of the automated tests I have been working on. You can login/register on my personal gitlab server with your github account.

## 1.16.6 Development

You will find any development information on the [developer guide](#) page.

## 1.17 Changelog

### 1.17.1 0.6.0 (05/14/2018)

- **BREAKING:** the *BASIC* ACL engine will now grant users on all agents if they are not explicitly defined
- **BREAKING:** a new `[ACL]` section has been created in order to control the new ACL engine behavior
- **BREAKING:** the *Burp1* and *Burp2* configuration sections have been merged into one single *Burp* section
- **BREAKING:** the *running* backups are now displayed in green instead of blue
- **BREAKING:** the docker postgresql image was upgraded from 9.6 to 10.1, you'll have to manually upgrade/migrate your data [following this documentation](#)
- **BREAKING:** the `docker-compose.yml` file now uses the `version: '2'` format
- **BREAKING:** the old config file format with colons (:) as separator is no
- Add: new plugins system to allow users to write their own modules
- Add: [Italian translation](#) thanks to Enrico
- Add: new [client configuration templates](#)
- Add: [backups deletion](#)
- Add: [show last client status in client view](#)
- Add: [record login failure attempt](#)
- Add: support new burp counters
- Add: support new burp pair options
- Add: support new reset list (:=) syntax
- Add: new websocket server
- Add: new [Administration panel](#)
- Improvement: better ACL engine
- Fix: issue #213
- Fix: issue #225
- Fix: issue #226
- Fix: issue #227
- Fix: issue #234
- Fix: issue #235

- Fix: [issue #236](#)
- Fix: [issue #242](#)
- Fix: [issue #245](#)
- Fix: [issue #246](#)
- Fix: [issue #247](#)
- Fix: [issue #248](#)
- Fix: [issue #251](#)
- Fix: [issue #257](#)
- Fix: [issue #262](#)
- Fix: [issue #263](#)
- Fix: [issue #264](#)
- [Full changelog](#)

### 1.17.2 0.5.1 (05/26/2017)

- Fix: handle non ascii chars in the browser view
- Fix: [issue #215](#)
- Fix: [issue #218](#)

### 1.17.3 0.5.0 (05/09/2017)

- **BREAKING:** the *standalone* option has been renamed to *single* for less confusion
- **BREAKING:** the `bui-agent` has now its own independent package to reduce dependencies
- Add: [Spanish translation](#) thanks to Pablo
- Add: [reverse\\_proxy](#) option while running through gunicorn
- Add: [OS detection](#) for burp 2
- Add: [customizable reports](#)
- Add: persistent storage for user preferences if SQL storage enabled
- Add: rate-limiting of the API
- Add: new `diag` and `sysinfo` commands to help you diagnose issues
- Improvement: the SQL storage will detect out-of-sync schemas and disable itself if needed (with a log message)
- Improvement: speedup the calendar view
- Improvement: more complete user panel ([#185](#), [#184](#), [#182](#), [#132](#))
- Fix: [issue #186](#)
- Fix: [issue #192](#)
- Fix: [issue #194](#)
- Fix: [issue #196](#)
- Fix: [issue #198](#)

- Fix: [issue #210](#)
- Various bugfix
- [Full changelog](#)

#### 1.17.4 0.4.4 (01/02/2017)

- Fix: [issue #193](#)

#### 1.17.5 0.4.3 (12/28/2016)

- Fix: [issue #186](#)
- Fix: [issue #188](#)
- Fix: [issue #190](#)
- Fix: missing configuration in docker image
- Fix: help troubleshooting some errors
- Fix: missing vss\_strip binary
- Fix: encoding error that made unable to browse backups with burp1 backend

#### 1.17.6 0.4.2 (12/16/2016)

- Fix: bui-agent was broken
- Fix: handle i18n exceptions
- Fix: enable db migration only when needed
- Fix: wrong escape in translation

#### 1.17.7 0.4.1 (12/15/2016)

- **BREAKING:** Use the new Flask's embedded server by default means no more SSL (HTTPS) support without a dedicated application server
- Fix: [issue #156](#)
- Fix: [issue #157](#)
- Fix: [issue #165](#)
- Fix: [issue #176](#)
- Fix: [issue #181](#)
- Fix: [issue #182](#)
- Various bugfix
- [Full changelog](#)

### 1.17.8 0.4.0 (11/23/2016)

- **BREAKING:** The database schema evolved. In order to apply these modifications, you **MUST** run the `bui-manage db upgrade` command after upgrading
- **BREAKING:** Plain text passwords are deprecated since v0.3.0 and are now disabled by default
- **BREAKING:** The default *version* setting has been set to 2 instead of 1
- Add: new `bui-manage setup_burp` command
- Add: new `docker` image
- Add: manage user sessions
- Add: `French` translation
- Fix: issue #151
- Fix: issue #154
- Fix: issue #158
- Fix: issue #163
- Fix: issue #164
- Fix: issue #166
- Fix: issue #169
- Fix: issue #171
- Fix: issue #172
- Fix: issue #173
- Fix: issue #174
- Various bugfix
- `Full changelog`

### 1.17.9 0.3.0 (08/15/2016)

- **BREAKING:** New configuration file format to allow further improvements (The conversion is automatic, but LDAP settings might need some attention)
- **BREAKING:** Passwords are now *salted* for the *BASIC* authentication backend (The conversion is automatic too)
- **BREAKING:** If you plan to use the SQL storage along with gunicorn, you **MUST** add the `-preload` parameter (see the `gunicorn.d/burp-ui` file)
- Add: `Celery` support for asynchronous tasks
- Add: `SQLAlchemy` support for persistent storage
- Add: `RESTful` restore
- Add: `autoreload` config
- Add: remember some user settings
- Add: `client certificate` revocation
- Add: new `local` authentication backend

- Add: new filters on history API call
- Add: implement backend `keepalive`
- Add: allow to *disable* server-initiated restoration
- Fix: disable Basic-Auth login from UI to prevent some bugs with sessions
- Fix: issue #134
- Fix: issue #135
- Fix: issue #137
- Fix: issue #138
- Fix: issue #145
- Fix: issue #148
- Improvement: new asynchronous `backup-running` API call
- Security: restrict files that can be sent by the agent
- [Full changelog](#)

### 1.17.10 0.2.1 (05/17/2016)

- Add: allow to edit a server-initiated restoration
- Add: allow to cancel a server-initiated restoration
- Add: support for Burp labels
- Add: `server-initiated` backups
- Add: support sub-root path
- Add: new Burp 2 settings
- Improvement: better logging system
- Improvement: new security options
- Fix: issue #109
- Fix: issue #113
- Fix: issue #114
- Fix: issue #117
- Fix: issue #123
- Doc
- [Full changelog](#)

### 1.17.11 0.1.3 (02/20/2016)

- Fix: issue #107
- Fix: issue #108

### 1.17.12 0.1.2 (02/18/2016)

- Fix: duration computation
- Fix: issue #104
- Fix: issue #105
- Fix: issue #106

### 1.17.13 0.1.1 (02/17/2016)

- Fix: burp2 backend issue
- Fix: Debian wheezy compatibility
- Fix: sample configuration files location
- Better calendar readability

### 1.17.14 0.1.0 (02/15/2016)

- Add: [python 3](#) support
- Add: new fields in [backup](#) reports
- Add: [server-side](#) initiated restoration
- Add: percent done in [overview](#)
- Add: ability to [chain multiple](#) authentication backends
- Add: display versions [within the interface](#)
- Add: support for [zip64](#)
- Add: new [report](#)
- Add: new [calendar view](#)
- Add: “restart” option to debian init script thanks to @Larsen
- Add: Basic HTTP Authentication (mostly for the API)
- Add: self-documented API
- Fix: issue #81
- Fix: issue #87
- Fix: issue #88
- Fix: issue #92
- Fix: issue #95
- Fix: issue #99
- Fix: issue #100
- Fix: issue #101
- [demo](#)
- API refactoring

- Security fixes
- Bugfixes

### **1.17.15 0.0.7.3 (09/26/2015)**

- Fix: issue #77
- Doc

### **1.17.16 0.0.7.2 (09/01/2015)**

- Fix: issue #73
- Fix: issue #74
- Doc

### **1.17.17 0.0.7.1 (08/22/2015)**

- Add: Burp-2 backend
- Add: sortable tables
- Add: ACL support
- Add: support client-side encrypted backups while performing an online restoration
- Add: multiple archive format
- Add: better Active Directory support
- Improvement: better config file parser
- Improvement: better logging with Unicorn
- Improvement: full support of server configuration file + clientconfdir
- Fix: issue #35
- Fix: issue #37
- Fix: issue #41
- Fix: issue #42
- Fix: issue #46
- Fix: issue #49
- Fix: issue #53
- Fix: issue #54
- Fix: issue #59
- Fix: issue #62
- Fix: issue #68
- Fix: issue #69
- Fix: issue #70



- Fix: issue #71
- Fix: issue #72
- doc on [readthedocs](#)
- Two merge requests from Wade Fitzpatrick (!1 and !2)
- API refactoring
- Security fixes
- Bugfixes
- [Full changelog](#)

#### **1.17.18 0.0.6 (12/15/2014)**

- Add: [gunicorn](#) support
- Add: init script for CentOS
- Add: init script for Debian
- Add: autofocus login field on login page
- Add: [burp-server](#) configuration panel
- Fix: issue #25
- Fix: issue #26
- Fix: issue #30
- Fix: issue #32
- Fix: issue #33
- Fix: issue #34
- Fix: issue #35
- Fix: issue #39
- Code cleanup
- Improve unit tests
- Bugfixes
- [Full changelog](#)

#### **1.17.19 0.0.5 (09/22/2014)**

- Add: multi-server support
- Fix bugs
- [Full changelog](#)

### 1.17.20 0.0.4 (09/07/2014)

- Add: ability to download files directly from the web interface
- [Full changelog](#)

### 1.17.21 0.0.3 (09/02/2014)

- Add: authentication
- [Full changelog](#)

### 1.17.22 0.0.2 (08/25/2014)

- Fix bugs
- [Full changelog](#)

### 1.17.23 0.0.1 (08/25/2014)

- Initial release

## 1.18 FAQ

### 1.18.1 Is there a demo somewhere?

Yes, you can play with [Burp-UI](#) at [demo.burp-ui.org](http://demo.burp-ui.org). Credentials are:

- *admin / admin* to play with [Burp-UI](#) as an administrator
- *moderator / moderator* to play with [Burp-UI](#) as a moderator
- *demo / demo* to play with [Burp-UI](#) as a regular user

### 1.18.2 How to start using Burp-UI?

You may find all the basic informations to get started with [Burp-UI](#) in the [README](#) file and in this documentation. You can also read the [step-by-step](#) page for some detailed use-cases.

### 1.18.3 How does Burp-UI work?

The answer depends whether you are using burp 1.x or burp 2.x. Basically, [Burp-UI](#) tries to provide a consistent API between the *Frontend* (the UI) and the burp server. To do so, it implements two *Backends*: burp-1 and burp-2. You can select either of these with the [version](#) flag in your configuration.

You can also refer to the [Architecture](#) page of the documentation to know more about those backends.

### 1.18.4 How to configure my *firewall*?

When running Burp-UI in single *mode*, the embedded webserver listens on port **5000** on all interfaces.

The Burp-UI agents listen on port **10000** by default.

Of course those are configurable.

### 1.18.5 What are the default credentials?

The default login / password is *admin / admin* with the *basic* authentication backend.

### 1.18.6 How does the online restoration feature work?

The online restoration feature works the same way as if you were running the burp client yourself. It means Burp-UI runs the following command:

```
burp -a r -b <number> -C <client name> -r <regex> -d /tmp/XXX -c <bconfcli>
```

It then generates an archive based on the restored files.

Because of this workflow, and especially the use of the `-C` flag you need to tell your burp-server the client used by Burp-UI can perform a restoration for a different client. You can refer to the *restoration* section of this documentation along with the *version* section for more details.

### 1.18.7 What does the server-initiated restoration feature do and how to make it work?

This feature asks the server to perform a restoration on the client the next time it sees it.

In order for this feature to work, your client **MUST** allows the server to do that. You have to set `server_can_restore = 1` (which is the default value) in your client configuration file (usually `/etc/burp/burp.conf`).

### 1.18.8 How can I start Burp-UI as a daemon?

There are several *init scripts* provided by some users available [here](#).

---

**Note:** I do not (and cannot) support these scripts. Only the *Gunicorn* way is supported.

---

The recommended way to run Burp-UI in production is to use *Gunicorn*. You can refer to the *gunicorn* section of this documentation for more details.

### 1.18.9 How to setup a reverse-proxy in front of Burp-UI?

The only way to run Burp-UI behind a reverse-proxy is to use *Gunicorn*. You can refer to the *gunicorn* section of this documentation for more details.

### 1.18.10 Why don't I see all my clients using the burp-2 backend?

Starting with burp 2, you cannot see all the client through the status port unless you tell burp a particular client can see other clients statistics. See the [general instructions](#) for more details.

### 1.18.11 Are there any known issues?

There is a [known issue](#) section in this documentation.

### 1.18.12 I cannot find the `bui-agent` command anymore, where is it?

Since *v0.5.0*, the `bui-agent` has it's own package in order to reduce requirements. The agent does not need the Flask requirements and so on. You can now install it with the `pip install burp-ui-agent` command. Alternatively, there is now a `bui-agent-legacy` provided by the `burp-ui` package.

See the [upgrading](#) section for more details.

### 1.18.13 Why using redis?

Redis may be used for several things:

- store the sessions server side (by default sessions are stored client side in a secure cookie)
- cache some data
- monitor API usage for the rate limiter

All of these features are totally optional. Redis is also used by celery to interact between Burp-UI and the asynchronous worker.

### 1.18.14 Why using SQL?

The SQL database is currently used to keep a track of several meta-data. Since *v0.5.0*, the SQL database is able to store user preferences. Again, it is totally optional to use it.

### 1.18.15 Why using Celery?

Celery is used to run some asynchronous jobs such as reports computations or online restorations.

Computing reports asynchronously allows faster answer especially when you manage several dozens of clients.

### 1.18.16 Burp-UI does not seem to understand the *bind* and *port* options anymore, what should I do?

Since *v0.4.0*, the new Flask development server is used when running in *single* mode. The *bind* and *port* options are not read anymore. You can either run **Burp-UI** with the `-- -h x.x.x.x -p yyyy` flags or use the legacy launcher `python -m burpui -m legacy [--help]`. See the [upgrading](#) page for details.

### 1.18.17 Burp-UI does not work anymore since I upgraded it, what can I do?

Make sure you read the [upgrading](#) page in case some breaking changes occurred.

### 1.18.18 I am getting errors while restoring large files (>3GB), what should I do?

The default *zip* module does not support large files by default. You can either enable large file support by setting `zip64 = true` in the [Experimental] section. Alternatively, you can choose an other compression module by selecting an other extension while proceeding the restoration.

### 1.18.19 I see a lot of *cannot spawn burp process* errors, what can I do?

This error means *Burp-UI* is not able to communicate with the burp server. You should check your logs (both *Burp-UI*'s and burp server's) to understand what is wrong. If you are using *Gunicorn*, it is possible you reached the limit of *status children*. You can safely increase the `max_status_children` setting in your *burp-server.conf* file to 15 (the default is 5). You can also check your *status port* is open and/or accessible by your client. To do so, you can run the `burp -a m` command.

### 1.18.20 How can I contribute?

You can refer to the [contributing](#) section of this documentation.

## 1.19 Step By Step

Although *Burp-UI* tries to make *Burp* accessible to everyone, both products have their complexity.

In this *Step by Step*, I would like to introduce you different use-cases with their associated configurations, descriptions and comments. In every case, we will consider neither *Burp* or *Burp-UI* are installed and describe the steps to setup your server from Scratch.

---

**Note:** Again, this part of the doc is mostly debian-centric. If some users are willing to adapt these examples with other distros I would be very thankful.

---

1. *Burp1 server* with *Burp-UI*
2. *Burp2 server* with *Burp-UI*

### 1.19.1 Burp1 server

In this scenario, we are going to install a *Burp* server version 1.4.40 which is the current stable version. We assume you are using the user *root* to run the following commands.

We begin with the installation of *Burp* itself.

First, we need some system requirements in order to compile *Burp* and to install *Burp-UI*:

```
apt-get update
apt-get install uthash-dev g++ make libssl-dev librsync-dev python2.7-dev \
git python-pip libffi-dev
```

Now we retrieve the *Burp* sources and then we compile and install it:

```
cd /usr/src
git clone https://github.com/grke/burp.git
cd burp
git checkout tags/1.4.40
./configure --disable-ipv6
make
make install
# we also install init scripts
cp debian/init /etc/init.d/burp
cat >/etc/default/burp<<EOF
RUN="yes"
DAEMON_ARGS="-c /etc/burp/burp-server.conf"
EOF
chmod +x /etc/init.d/burp
update-rc.d burp defaults
```

It is now time to install **Burp-UI**:

```
pip install --upgrade burp-ui
```

Now that everything is installed, let's configure our tools!

In order to perform online restorations, **Burp-UI** relies on a classical **Burp** client.

We need to define our client, and we also need to allow it to perform restorations for other clients. We will set it up globally. Our client will be named *bui*:

```
# burp-ui client's definition
cat >/etc/burp/clientconfdir/bui<<EOF
password = abcdefgh
EOF

# grant our client to perform restorations for others
echo "restore_client = bui" >>/etc/burp/burp-server.conf

# now we generate ou client configuration
cat >/etc/burp/burp.conf<<EOF
mode = client
port = 4971
server = 127.0.0.1
password = abcdefgh
cname = bui
pidfile = /var/run/burp.bui.pid
syslog = 0
stdout = 1
progress_counter = 1
ca_burp_ca = /usr/sbin/burp_ca
ca_csr_dir = /etc/burp/CA-client
# SSL certificate authority - same file on both server and client
ssl_cert_ca = /etc/burp/ssl_cert_ca-client.pem
# Client SSL certificate
ssl_cert = /etc/burp/ssl_cert-client.pem
# Client SSL key
ssl_key = /etc/burp/ssl_cert-client.key
# SSL key password
ssl_key_password = password
# Common name in the certificate that the server gives us
ssl_peer_cn = burpserver
```

(continues on next page)

(continued from previous page)

```
# The following options specify exactly what to backup.
include = /home
EOF
```

Our **Burp** server is now set up, we can start it:

```
/etc/init.d/burp start
```

Now we can configure **Burp-UI**. The package comes with a default configuration and init scripts. We copy them at the right place:

```
cp /usr/local/share/burpui/contrib/debian/init.sh /etc/init.d/burp-ui
chmod +x /etc/init.d/burp-ui
update-rc.d burp-ui defaults
cp /usr/local/share/burpui/etc/burpui.sample.cfg /etc/burp/burpui.cfg
```

We have to edit the default configuration in order to work with a **Burp-1** server:

```
sed -i "s/^version = .*/version = 1/" /etc/burp/burpui.cfg
```

The default configuration is plug and play for this case, we just have to start **Burp-UI**:

```
/etc/init.d/burp-ui start
```

Your server is now fully set-up, you can access **Burp-UI** by pointing your browser to: [http://server\\_ip:5000/](http://server_ip:5000/)

The default user / password is: admin / admin

For further customization, you can refer to the [usage](#) page of this documentation.

## 1.19.2 Burp2 server

In this scenario, we are going to install a **Burp** server version 2.0.54. We assume you are using the user *root* to run the following commands.

We begin with the installation of **Burp** itself.

First, we need some system requirements in order to compile **Burp** and to install **Burp-UI**:

```
apt-get update
apt-get install uthash-dev g++ make libssl-dev librsync-dev python2.7-dev \
    git python-pip libffi-dev libyajl-dev libz-dev
```

Now we retrieve the **Burp** sources and then we compile and install it:

---

**Note:** There are now plenty of official and unofficial burp packages. See [here](#) for details.

---

```
cd /usr/src
git clone https://github.com/grke/burp.git
cd burp
git checkout tags/2.0.54
./configure --prefix=/usr --sysconfdir=/etc/burp --localstatedir=/var
make
make install
```

(continues on next page)

(continued from previous page)

```
# we also install init scripts
cp debian/init /etc/init.d/burp
cat >/etc/default/burp<<EOF
RUN="yes"
DAEMON_ARGS="-c /etc/burp/burp-server.conf"
EOF
chmod +x /etc/init.d/burp
update-rc.d burp defaults
```

It is now time to install **Burp-UI**:

```
pip install --upgrade burp-ui
```

Now that everything is installed, let's configure our tools!

In order to perform online restorations, **Burp-UI** relies on a classical **Burp** client.

We need to define our client, and we also need to allow it to perform restorations for other clients. We will set it up globally. Our client will be named *bui*:

```
# burp-ui client's definition
cat >/etc/burp/clientconfdir/bui<<EOF
password = abcdefgh
EOF

# grant our client to perform restorations for others
echo "restore_client = bui" >>/etc/burp/burp-server.conf
# Burp 2 is able to cache the manifests for better performances
echo "monitor_browse_cache = 1" >>/etc/burp/burp-server.conf

# now we generate ou client configuration
cat >/etc/burp/burp.conf<<EOF
mode = client
port = 4971
status_port = 4972
server = ::1
password = abcdefgh
cname = bui
pidfile = /var/run/burp.bui.pid
syslog = 0
stdout = 1
progress_counter = 1
network_timeout = 72000
ca_burp_ca = /usr/sbin/burp_ca
ca_csr_dir = /etc/burp/CA-client
# SSL certificate authority - same file on both server and client
ssl_cert_ca = /etc/burp/ssl_cert_ca-client.pem
# Client SSL certificate
ssl_cert = /etc/burp/ssl_cert-client.pem
# Client SSL key
ssl_key = /etc/burp/ssl_cert-client.key
# SSL key password
ssl_key_password = password
# Common name in the certificate that the server gives us
ssl_peer_cn = burpserver
# The following options specify exactly what to backup.
include = /home
EOF
```



Our **Burp** server is now set up, we can start it:

```
/etc/init.d/burp start
```

Now we can configure **Burp-UI**. The package comes with a default configuration and init scripts. We copy them at the right place:

```
cp /usr/local/share/burpui/contrib/debian/init.sh /etc/init.d/burp-ui
chmod +x /etc/init.d/burp-ui
update-rc.d burp-ui defaults
cp /usr/local/share/burpui/etc/burpui.sample.cfg /etc/burp/burpui.cfg
```

That's it, the other default parameter should be able to handle such a setup. We can start **Burp-UI**:

```
/etc/init.d/burp-ui start
```

Your server is now fully set-up, you can access **Burp-UI** by pointing your browser to: [http://server\\_ip:5000/](http://server_ip:5000/)

The default user / password is: admin / admin

For further customization, you can refer to the [usage](#) page of this documentation.

## 1.20 Developer Guide

### 1.20.1 Development

If you wish to use the latest and yet unstable version (eg. [master](#)), you can install it using `pip` too, but I would recommend you to use a `virtualenv`.

To do so, run the following commands:

```
mkdir /opt/bui-venv
pip install virtualenv
virtualenv /opt/bui-venv
source /opt/bui-venv/bin/activate
cd /tmp
# the .tar.gz works with both python 2 and python 3
wget https://git.ziirish.me/ziirish/burp-ui/builds/artifacts/master/download?
  ↪ job=build:py2 -O burp-ui_build.zip
unzip burp-ui_build.zip
pip install --upgrade dist/burp-ui-*.tar.gz
# Now if you want to test the latest bui-agent run this command:
#pip install --upgrade meta/burp-ui-agent-*.tar.gz
```

You can uninstall/disable this **Burp-UI** setup by typing `deactivate` and removing the `/opt/bui-venv` directory.

### 1.20.2 Hacking

For those of you who would like to hack on the project, I have split out the repository to keep a copy of all the external dependencies (JS and CSS) in a git submodule.

In order to run local debugging, you need to retrieve this git submodule.

To do so, run the following commands:

```
git clone https://git.ziirish.me/ziirish/burp-ui.git
cd burp-ui
git submodule update --init
```

Before submitting your code, make sure the tests still run. To do that, you can use `tox` like this:

```
pip install tox
tox
```

By defaults, it will run tests against python 2.7, 3.4 and 3.6. However, you can choose the versions specifically like this:

```
tox -e py27
```

### 1.20.3 create\_app

The `create_app` method is the most important. This is where the application is created. It is mostly used when running your application through `gunicorn`.

`burpui.create_app(conf=None, verbose=0, logfile=None, **kwargs)`  
Initialize the whole application.

#### Parameters

- **conf** (*str*) – Configuration file to use
- **verbose** (*int*) – Set the verbosity level
- **logfile** (*str*) – Store the logs in the given file
- **kwargs** (*dict*) – Extra options: - `gunicorn` (bool): Enable gunicorn engine instead of flask's default. Default is True. - `unittest` (bool): Are we running tests (used for test only). Default is False. - `debug` (bool): Enable debug mode. Default is False. - `cli` (bool): Are we running the CLI. Default is False. - `reverse_proxy` (bool): Are we behind a reverse-proxy. Default is True if gunicorn is True

**Returns** A `burpui.server.BUIServer` object

### 1.20.4 API

Here are the different routes provided by the application. You can implement whatever front-end you like on top of it.

The API supports HTTP Basic authentication through the *Authorization* Header. By default, each authorization is only valid for **one** request (ie. the sessions are automatically revoked after each request complete). You can ask for reusable sessions though. Here are the HTTP headers supported:

- `X-Reuse-Session`: set it to `true` to be able to reuse sessions
- `X-Language`: set it to whatever supported language you want

Don't forget to call `/logout` once you're done if you choose to use reusable sessions.

#### **GET /api/admin/auth/backends**

Returns a list of backends

**GET** method provided by the webservice.

**Returns** Backends

**DELETE** /api/admin/auth/ (*backend*) /users/  
*name*

**DELETE** /api/admin/auth/ (*backend*) /users

**DELETE** /api/admin/auth/users/ (*name*)

**DELETE** /api/admin/auth/users

Delete a user

**GET** /api/admin/auth/ (*backend*) /users/  
*name*

**GET** /api/admin/auth/ (*backend*) /users

**GET** /api/admin/auth/users/ (*name*)

**GET** /api/admin/auth/users

Returns a list of users

**GET** method provided by the webservice.

**Returns** Users

**PUT** /api/admin/auth/ (*backend*) /users/  
*name*

**PUT** /api/admin/auth/ (*backend*) /users

**PUT** /api/admin/auth/users/ (*name*)

**PUT** /api/admin/auth/users

Create a new user

**POST** /api/admin/auth/ (*backend*) /users/  
*name*

**POST** /api/admin/auth/ (*backend*) /users

**POST** /api/admin/auth/users/ (*name*)

**POST** /api/admin/auth/users

Change user password

**GET** /api/admin/acl/ (*backend*) /moderators

**GET** /api/admin/acl/moderators

Returns a list of moderators

**GET** method provided by the webservice.

**Returns** Moderators

**DELETE** /api/admin/acl/ (*backend*) /moderator/  
*member*

**DELETE** /api/admin/acl/ (*backend*) /moderator

**DELETE** /api/admin/acl/moderator

Remove a moderator member

**DELETE** method provided by the webservice.

**GET** /api/admin/acl/ (*backend*) /moderator/  
*member*

**GET** /api/admin/acl/ (*backend*) /moderator

**GET /api/admin/acl/moderator**

Returns a list of moderator users

GET method provided by the webservice.

**Returns** Members

**PUT /api/admin/acl/ (backend) /moderator/  
member**

**PUT /api/admin/acl/ (backend) /moderator**

**PUT /api/admin/acl/moderator**

Add a member as moderator

PUT method provided by the webservice.

**POST /api/admin/acl/ (backend) /moderator/  
member**

**POST /api/admin/acl/ (backend) /moderator**

**POST /api/admin/acl/moderator**

Update moderator grants

POST method provided by the webservice.

**GET /api/admin/acl/backends**

Returns a list of backends

GET method provided by the webservice.

**Returns** Backends

**GET /api/admin/acl/ (backend) /admins**

**GET /api/admin/acl/admins**

Returns a list of admins

GET method provided by the webservice.

**Returns** Moderators

**DELETE /api/admin/acl/ (backend) /groups/  
name**

**DELETE /api/admin/acl/ (backend) /groups**

**DELETE /api/admin/acl/groups/ (name)**

**DELETE /api/admin/acl/groups**

Delete a group

**GET /api/admin/acl/ (backend) /groups/  
name**

**GET /api/admin/acl/ (backend) /groups**

**GET /api/admin/acl/groups/ (name)**

**GET /api/admin/acl/groups**

Returns a list of group

GET method provided by the webservice.

**Returns** Groups

**PUT** `/api/admin/acl/ (backend) /groups/`  
*name*

**PUT** `/api/admin/acl/ (backend) /groups`

**PUT** `/api/admin/acl/groups/ (name)`

**PUT** `/api/admin/acl/groups`  
 Create a new group

**POST** `/api/admin/acl/ (backend) /groups/`  
*name*

**POST** `/api/admin/acl/ (backend) /groups`

**POST** `/api/admin/acl/groups/ (name)`

**POST** `/api/admin/acl/groups`  
 Change a group

**DELETE** `/api/admin/acl/ (backend) /grants/`  
*name*

**DELETE** `/api/admin/acl/ (backend) /grants`

**DELETE** `/api/admin/acl/grants/ (name)`

**DELETE** `/api/admin/acl/grants`  
 Delete a grant

**GET** `/api/admin/acl/ (backend) /grants/`  
*name*

**GET** `/api/admin/acl/ (backend) /grants`

**GET** `/api/admin/acl/grants/ (name)`

**GET** `/api/admin/acl/grants`  
 Returns a list of grants

**GET** method provided by the webservice.

**Returns** Grants

**PUT** `/api/admin/acl/ (backend) /grants/`  
*name*

**PUT** `/api/admin/acl/ (backend) /grants`

**PUT** `/api/admin/acl/grants/ (name)`

**PUT** `/api/admin/acl/grants`  
 Create a new grant

**POST** `/api/admin/acl/ (backend) /grants/`  
*name*

**POST** `/api/admin/acl/ (backend) /grants`

**POST** `/api/admin/acl/grants/ (name)`

**POST** `/api/admin/acl/grants`  
 Change a grant

**DELETE** `/api/admin/acl/ (backend) /admin/`  
*member*

**DELETE** /api/admin/acl/ (backend) /admin

**DELETE** /api/admin/acl/admin

Remove an admin member

**DELETE** method provided by the webservice.

**GET** /api/admin/acl/ (backend) /admin/  
member

**GET** /api/admin/acl/ (backend) /admin

**GET** /api/admin/acl/admin

Returns a list of admin users

**GET** method provided by the webservice.

**Returns** Members

**PUT** /api/admin/acl/ (backend) /admin/  
member

**PUT** /api/admin/acl/ (backend) /admin

**PUT** /api/admin/acl/admin

Add a member as admin

**PUT** method provided by the webservice.

**DELETE** /api/admin/acl/ (backend) /group/  
name/member

**DELETE** /api/admin/acl/ (backend) /group/  
name

**DELETE** /api/admin/acl/group

Remove a member from a given group

**DELETE** method provided by the webservice.

**GET** /api/admin/acl/ (backend) /group/  
name/member

**GET** /api/admin/acl/ (backend) /group/  
name

**GET** /api/admin/acl/group

Returns a list of users in a giver group

**GET** method provided by the webservice.

**Returns** Members

**PUT** /api/admin/acl/ (backend) /group/  
name/member

**PUT** /api/admin/acl/ (backend) /group/  
name

**PUT** /api/admin/acl/group

Add a member in a given group

**PUT** method provided by the webservice.

**GET** /api/admin/me/session/ (uuid: id)

**GET /api/admin/me/session**

Returns a list of sessions

**GET** method provided by the webservice.

**Returns** Sessions

**DELETE /api/admin/me/session/ (uuid: id)**

**DELETE /api/admin/me/session**

Delete a given session

Note: `id` is mandatory

**DELETE /api/preferences/ui**

Delete prefs

**GET /api/preferences/ui**

Returns a list of prefs

**GET** method provided by the webservice.

**Returns** prefs

**PUT /api/preferences/ui**

Create prefs

**POST /api/preferences/ui**

Change prefs

**GET /api/settings/ (server) /server-config/  
path: conf**

**GET /api/settings/ (server) /server-config**

**GET /api/settings/server-config/ (path: conf)**

**GET /api/settings/server-config**

Reads the server configuration

**GET** method provided by the webservice.

The *JSON* returned is:

```
{
  "boolean": [
    "daemon",
    "fork",
    "..."
  ],
  "defaults": {
    "address": "",
    "autoupgrade_dir": "",
    "ca_burp_ca": "",
    "ca_conf": "",
    "ca_name": "",
    "ca_server_name": "",
    "client_can_delete": true,
    "...": "..."
  },
  "integer": [
    "port",
    "status_port",
  ]
}
```

(continues on next page)

(continued from previous page)

```

    "...",
  ],
  "multi": [
    "keep",
    "restore_client",
    "...",
  ],
  "placeholders": {
    "autoupgrade_dir": "path",
    "ca_burp_ca": "path",
    "ca_conf": "path",
    "ca_name": "name",
    "ca_server_name": "name",
    "client_can_delete": "0|1",
    "...": "...",
  },
  "results": {
    "boolean": [
      {
        "name": "hardlinked_archive",
        "value": false,
        "reset": false
      },
      {
        "name": "syslog",
        "value": true,
        "reset": false
      },
      { "...": "..." }
    ],
    "clients": [
      {
        "name": "testclient",
        "value": "/etc/burp/clientconffdir/testclient"
      }
    ],
    "common": [
      {
        "name": "mode",
        "value": "server",
        "reset": false
      },
      {
        "name": "directory",
        "value": "/var/spool/burp",
        "reset": false
      },
      { "...": "..." }
    ],
    "includes": [],
    "includes_ext": [],
    "integer": [
      {
        "name": "port",
        "value": 4971,
        "reset": false
      },
    ],
  },

```

(continues on next page)



(continued from previous page)

```

    {
      "name": "status_port",
      "value": 4972,
      "reset": false
    },
    { "...": "..." }
  ],
  "multi": [
    {
      "name": "keep",
      "value": [
        "7",
        "4"
      ],
      "reset": [
        false,
        true
      ]
    },
    { "...": "..." }
  ],
  "hierarchy": [
    {
      "children": [
        {
          "children": [],
          "dir": "/tmp/burp/conf.d",
          "full": "/tmp/burp/conf.d/empty.conf",
          "name": "empty.conf",
          "parent": "/tmp/burp/burp-server.conf"
        },
        {
          "children": [],
          "dir": "/tmp/burp/conf.d",
          "full": "/tmp/burp/conf.d/ipv4.conf",
          "name": "ipv4.conf",
          "parent": "/tmp/burp/burp-server.conf"
        }
      ],
      "dir": "/tmp/burp",
      "full": "/tmp/burp/burp-server.conf",
      "name": "burp-server.conf",
      "parent": null
    }
  ],
  "server_doc": {
    "address": "Defines the main TCP address that the server listens on. The
    ↪ default is either ':::' or '0.0.0.0', dependent upon compile time options.",
    "...": "..."
  },
  "string": [
    "mode",
    "address",
    "..."
  ],
  "suggest": {

```

(continues on next page)

(continued from previous page)

```
"compression": [
  "gzip1",
  "gzip2",
  "gzip3",
  "gzip4",
  "gzip5",
  "gzip6",
  "gzip7",
  "gzip8",
  "gzip9"
],
"mode": [
  "client",
  "server"
],
"...": []
}
}
```

### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

**Returns** The *JSON* described above.

**DELETE** /api/settings/ (*server*) /server-config/  
path: *conf*

**DELETE** /api/settings/ (*server*) /server-config

**DELETE** /api/settings/server-config/ (path: *conf*)

**DELETE** /api/settings/server-config

Deletes a configuration file

**POST** /api/settings/ (*server*) /server-config/

path: *conf*

**POST** /api/settings/ (*server*) /server-config

**POST** /api/settings/server-config/ (path: *conf*)

**POST** /api/settings/server-config

Saves the server configuration

**GET** /api/settings/ (*server*) /path-expander/

*client*

**GET** /api/settings/ (*server*) /path-expander

**GET** /api/settings/path-expander/ (*client*)

**GET** /api/settings/path-expander

Expends a given path

For instance if it's given a glob expression it will returns a list of files matching the expression.

**GET** /api/settings/ (*server*) /templates

**GET** /api/settings/templates

Returns a list of clients

**PUT** `/api/settings/ (server) /template`

**PUT** `/api/settings/template`

Creates a new template

**GET** `/api/settings/ (server) /clients`

**GET** `/api/settings/clients`

Returns a list of clients

**GET** `/api/settings/ (server) /options`

**GET** `/api/settings/options`

Returns various setting options

**PUT** `/api/settings/ (server) /config`

**PUT** `/api/settings/config`

Creates a new client

**GET** `/api/clients/ (server) /backup-running`

**GET** `/api/clients/backup-running`

Tells if a backup is running right now

**GET** method provided by the webservice.

The *JSON* returned is:

```
{
  "running": false
}
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

**Returns** The *JSON* described above.

**GET** `/api/clients/ (server) /running/ client`

**GET** `/api/clients/ (server) /running`

**GET** `/api/clients/running/ (client)`

**GET** `/api/clients/running`

Returns a list of clients currently running a backup

**GET** method provided by the webservice.

The *JSON* returned is:

```
[ 'client1', 'client2' ]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to see.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **client** (*str*) – Ask a specific client in order to know if it is running a backup

**Returns** The *JSON* described above.

**GET** `/api/clients/ (server) /report`

**GET** `/api/clients/report`

Returns a global report about all the clients of a given server

**GET** method provided by the webservice.

The *JSON* returned is:

```
{
  "backups": [
    {
      "name": "client1",
      "number": 15
    },
    {
      "name": "client2",
      "number": 1
    }
  ],
  "clients": [
    {
      "name": "client1",
      "stats": {
        "total": 296377,
        "totsize": 57055793698,
        "os": "unknown"
      }
    },
    {
      "name": "client2",
      "stats": {
        "total": 3117,
        "totsize": 5345361,
        "os": "windows"
      }
    }
  ]
}
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

**Returns** The *JSON* described above

**GET** `/api/servers/report`

Returns a global report about all the servers managed by Burp-UI

**GET** method provided by the webservice.

The *JSON* returned is:

```
{
  "backups": [
    {
```

(continues on next page)

(continued from previous page)

```

    "name": "AGENT1",
    "number": 49
  },
],
"servers": [
  {
    "name": "AGENT1",
    "number": {
      "os": {
        "linux": 4
      },
      "total": 349705,
      "totsize": 119400711726,
    }
  }
]
}

```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients/servers you are authorized to.

**Returns** The *JSON* described above.

**GET** `/api/clients/ (server) /stats`

**GET** `/api/clients/stats`

Returns a list of clients with their states

**GET** method provided by the webservice.

The *JSON* returned is:

```

{
  [
    {
      "last": "2015-05-17 11:40:02",
      "name": "client1",
      "state": "idle",
      "phase": "phase1",
      "percent": 12,
      "labels": []
    },
    {
      "last": "never",
      "name": "client2",
      "state": "idle",
      "phase": "phase2",
      "percent": 42,
      "labels": []
    }
  ]
}

```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

**Returns** The *JSON* described above

**GET** `/api/servers/stats`

Returns a list of servers (agents) with basic stats

**GET** method provided by the webservice.

The *JSON* returned is:

```
[
  {
    'alive': true,
    'clients': 2,
    'name': 'burp1',
  },
  {
    'alive': false,
    'clients': 0,
    'name': 'burp2',
  },
]
```

**Returns** The *JSON* described above.

**GET** `/api/clients/ (server) /all`

**GET** `/api/clients/all`

Returns a list of all clients with their associated Agent if any

**GET** method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "name": "client1",
    "agent": "agent1"
  },
  {
    "name": "client2",
    "agent": "agent1"
  },
  {
    "name": "client3",
    "agent": "agent2"
  }
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

**Returns** The *JSON* described above

**GET** `/api/client/ (server) /running/`  
*name*

**GET** `/api/client/running/ (name)`

**GET /api/client/ (*server*) /running**

**GET /api/client/running**

Returns the running status of a given client

**GET** method provided by the webservice.

The *JSON* returned is:

```
{
  "state": "running",
  "percent": 42,
  "phase": "2",
  "last": "now"
}
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

**Returns** The *JSON* described above.

**GET /api/admin/me**

Returns the current user informations

**GET** method provided by the webservice.

**Returns** User

**GET /api/misc/languages**

Returns a list of supported languages

**GET** method provided by the webservice.

The *JSON* returned is:

```
{
  "en": "English",
  "fr": "Français"
}
```

**Returns** The *JSON* described above.

**GET /api/misc/ (*server*) /counters/  
client**

**GET /api/misc/ (*server*) /counters**

**GET /api/misc/counters/ (*client*)**

**GET /api/misc/counters**

Returns counters for a given client

**GET** method provided by the webservice.

#### Parameters

- **name** – the client name if any. You can also use the GET parameter

‘name’ to achieve the same thing

**Returns** Counters**GET** `/api/misc/ (server) /monitor`**GET** `/api/misc/monitor`

Returns a list of clients that are currently running a backup

**GET** method provided by the webservice.The *JSON* returned is:

```
[
  {
    'client': 'client1',
    'agent': 'burp1',
    'counters': {
      'phase': 2,
      'path': '/etc/some/configuration',
      '...': '...',
    }
  },
  {
    'client': 'client12',
    'agent': 'burp2',
    'counters': {
      'phase': 3,
      'path': '/etc/some/other/configuration',
      '...': '...',
    }
  }
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

**Parameters**

- **server** (*str*) – Which server to collect data from when in multi-agent mode

**Returns** The *JSON* described above**GET** `/api/misc/ (server) /history/  
client`**GET** `/api/misc/history/ (client)`**GET** `/api/misc/ (server) /history`**GET** `/api/misc/history`

Returns a list of calendars describing the backups that have been completed so far

**GET** method provided by the webservice.The *JSON* returned is:

```
[
  {
    "color": "#7C6F44",
    "events": [
      {
        "backup": "0000001",
        "end": "2015-01-25 13:32:04+01:00",
```

(continues on next page)



(continued from previous page)

```

        "name": "toto-test",
        "start": "2015-01-25 13:32:00+01:00",
        "title": "Client: toto-test, Backup n°0000001",
        "url": "/client/toto-test"
    }
],
    "name": "toto-test",
    "textColor": "white"
}
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **client** (*str*) – Which client to collect data from

**Returns** The *JSON* described above

#### **POST** /api/misc/alert

Propagate a message to the next screen (or whatever reads the session)

#### **GET** /api/misc/ (*server*) /about

#### **GET** /api/misc/about

Returns various informations about Burp-UI

#### **GET** /api/swagger.json

Render the Swagger specifications as JSON

#### **GET** /api/doc

Override this method to customize the documentation page

#### **GET** /api/admin/auth/backend/ (*backend*)

Returns a given authentication backend

**GET** method provided by the webservice.

**Returns** Backend

#### **GET** /api/admin/acl/ (*backend*) /isModerator/ *member*

#### **GET** /api/admin/acl/isModerator/ (*member*)

Checks if a given member is moderator

#### **GET** /api/admin/acl/groupsOf/ (*member*)

Returns a list of group

**GET** method provided by the webservice.

**Returns** Groups

#### **GET** /api/admin/acl/ (*backend*) /isAdmin/ *member*

#### **GET** /api/admin/acl/isAdmin/ (*member*)

Checks if a given member is admin

**GET** `/api/admin/acl/backend/` (*backend*)  
Returns a given ACL backend

GET method provided by the webservice.

**Returns** Backend

**GET** `/api/settings/` (*server*) `/config/`  
*client/path:* *conf*

**GET** `/api/settings/config/` (*client*) `/`  
*path:* *conf*

**GET** `/api/settings/` (*server*) `/config/`  
*client*

**GET** `/api/settings/config/` (*client*)  
Reads a given client configuration

**DELETE** `/api/settings/` (*server*) `/config/`  
*client/path:* *conf*

**DELETE** `/api/settings/config/` (*client*) `/`  
*path:* *conf*

**DELETE** `/api/settings/` (*server*) `/config/`  
*client*

**DELETE** `/api/settings/config/` (*client*)  
Deletes a given client

**POST** `/api/settings/` (*server*) `/config/`  
*client/path:* *conf*

**POST** `/api/settings/config/` (*client*) `/`  
*path:* *conf*

**POST** `/api/settings/` (*server*) `/config/`  
*client*

**POST** `/api/settings/config/` (*client*)  
Saves a given client configuration

**PUT** `/api/restore/` (*server*) `/server-restore/`  
*name/int:* *backup*

**PUT** `/api/restore/server-restore/` (*name*) `/`  
*int:* *backup* Schedule a server-initiated restoration

PUT method provided by the webservice.

**Parameters**

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

**Returns** Status message (success or failure)

**GET** `/api/restore/` (*server*) `/server-restore/`  
*name*

**GET** `/api/restore/server-restore/ (name)`

Reads the content of the 'restore' file if present

**GET** method provided by the webservice.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

**Returns** The content of the restore file

**DELETE** `/api/restore/ (server) /server-restore/`  
*name*

**DELETE** `/api/restore/server-restore/ (name)`

Remove the 'restore' file if present

**DELETE** method provided by the webservice.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

**Returns** Status message (success or failure)

**POST** `/api/restore/ (server) /archive/`  
*name/int: backup*

**POST** `/api/restore/archive/ (name) /`  
*int: backup* Performs an online restoration

**POST** method provided by the webservice. This method returns a `flask.Response` object.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

**Returns** A `flask.Response` object representing an archive of the restored files

**PUT** `/api/backup/ (server) /do-server-backup/`  
*name*

**PUT** `/api/backup/do-server-backup/ (name)`

Schedule a server-initiated backup

**PUT** method provided by the webservice.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

**Returns** Status message (success or failure)

**GET** `/api/backup/ (server) /server-backup/`  
*name*

**GET** `/api/backup/server-backup/ (name)`

Tells if a 'backup' file is present

**GET** method provided by the webservice.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

**Returns** True if the file is found

**DELETE** `/api/backup/ (server) /server-backup/`  
*name*

**DELETE** `/api/backup/server-backup/ (name)`

Remove the 'backup' file if present

**DELETE** method provided by the webservice.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

**Returns** Status message (success or failure)

**GET** `/api/client/ (server) /browseall/`  
*name/int: backup*

**GET** `/api/client/browseall/ (name) /`  
**int:** *backup* Returns a list of all 'nodes' of a given backup

**GET** method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "date": "2015-05-21 14:54:49",
    "gid": "0",
    "inodes": "173",
    "selected": false,
    "expanded": false,
    "children": [],
    "mode": "drwxr-xr-x",
    "name": "/",
    "key": "/",
    "title": "/",
    "fullname": "/",
    "parent": "",
    "size": "12.0KiB",
    "type": "d",
    "uid": "0"
  }
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

**Returns** The *JSON* described above.

**GET** `/api/client/ (server) /browse/`  
*name/int: backup*

**GET** `/api/client/browse/ (name) /`  
*int: backup* Returns a list of ‘nodes’ under a given path

**GET** method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "date": "2015-05-21 14:54:49",
    "gid": "0",
    "inodes": "173",
    "selected": false,
    "expanded": false,
    "children": [],
    "mode": "drwxr-xr-x",
    "name": "/",
    "key": "/",
    "title": "/",
    "fullname": "/",
    "parent": "",
    "size": "12.0KiB",
    "type": "d",
    "uid": "0"
  }
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

**Returns** The *JSON* described above.

**GET** `/api/client/ (server) /report/`  
*name/int: backup*

**GET** `/api/client/ (server) /report/`  
*name*

**GET** `/api/client/report/ (name) /`  
*int: backup*

**GET** `/api/client/report/ (name)`  
 Returns a global report of a given backup/client

**GET** method provided by the webservice.

The *JSON* returned is:

```
{
  "dir": {
    "changed": 0,
    "deleted": 0,
    "new": 394,
    "scanned": 394,
    "total": 394,
    "unchanged": 0
  },
  "duration": 5,
  "efs": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "encrypted": true,
  "end": 1422189124,
  "files": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "files_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 1421,
    "scanned": 1421,
    "total": 1421,
    "unchanged": 0
  },
  "hardlink": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "meta": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "meta_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
```

(continues on next page)

(continued from previous page)

```

    "total": 0,
    "unchanged": 0
  },
  "number": 1,
  "received": 1679304,
  "softlink": {
    "changed": 0,
    "deleted": 0,
    "new": 1302,
    "scanned": 1302,
    "total": 1302,
    "unchanged": 0
  },
  "special": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "start": 1422189119,
  "total": {
    "changed": 0,
    "deleted": 0,
    "new": 3117,
    "scanned": 3117,
    "total": 3117,
    "unchanged": 0
  },
  "totsize": 5345361,
  "vssfooter": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssfooter_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssheader": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssheader_enc": {
    "changed": 0,

```

(continues on next page)

(continued from previous page)

```
"deleted": 0,
"new": 0,
"scanned": 0,
"total": 0,
"unchanged": 0
},
"windows": "false"
}
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

**Returns** The *JSON* described above.

**DELETE** `/api/client/ (server) /report/`  
*name/int: backup*

**DELETE** `/api/client/ (server) /report/`  
*name*

**DELETE** `/api/client/report/ (name) /`  
*int: backup*

**DELETE** `/api/client/report/ (name)`  
Deletes a given backup from the server

**DELETE** method provided by the webservice.

The access is filtered by the `burpui.misc.acl` module so that you can only delete backups you have access to.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

**GET** `/api/client/ (server) /labels/`  
*name*

**GET** `/api/client/labels/ (name)`  
Returns the labels of a given client

**GET** method provided by the webservice.

The *JSON* returned is:

```
{
  "labels": [
    "label1",
    "label2"
  ]
}
```



The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

**Returns** The *JSON* described above.

**GET** `/api/client/ (server) /stats/`  
*name*

**GET** `/api/client/stats/ (name)`  
Returns a list of backups for a given client

**GET** method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "date": "2015-01-25 13:32:00",
    "deletable": true,
    "encrypted": true,
    "received": 123,
    "size": 1234,
    "number": 1
  },
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

#### Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

**Returns** The *JSON* described above.

**GET** `/api/admin/session/ (user) /`  
*uuid: id*

**GET** `/api/admin/session/ (user)`  
Returns a list of sessions

**GET** method provided by the webservice.

**Returns** Sessions

## 1.20.5 Backend

Here is the *backend* interface definition in order to implement a new backend.

**class** `burpui.misc.backend.interface.BUIbackend (server=None, conf=None)`

The `burpui.misc.backend.interface.BUIbackend` class provides a consistent interface backend for any burp server.

#### Parameters

- **server** (`burpui.server.BUIServer`) – Flask server instance in order to access logger and/or some global settings
- **conf** (`str`) – Configuration file to use

**cancel\_server\_backup** (`client=None, agent=None`)

The `burpui.misc.backend.interface.BUIbackend.cancel_server_backup()` function is used to delete the server-initiated backup file of a given client.

**Parameters**

- **client** (`str`) – The name of the client to look for
- **agent** (`str`) – What server to ask (only in multi-agent mode)

**Returns** A list of notifications to return to the UI (success or failure)

Example:

```
[[0, "Success"]]
```

**cancel\_server\_restore** (`client=None, agent=None`)

The `burpui.misc.backend.interface.BUIbackend.cancel_server_restore()` function is used to delete the server-initiated restoration file of a given client.

**Parameters**

- **client** (`str`) – The name of the client to look for
- **agent** (`str`) – What server to ask (only in multi-agent mode)

**Returns** A list of notifications to return to the UI (success or failure)

Example:

```
[[0, "Success"]]
```

**clients\_list** (`agent=None`)

The `burpui.misc.backend.interface.BUIbackend.clients_list()` function is used to retrieve a list of clients with their configuration file.

**Returns** A list of clients with their configuration file

**del\_file** (`path, agent=None`)

The `burpui.misc.backend.interface.BUIbackend.del_file()` function is used to delete a file on a remote agent.

**delete\_backup** (`name=None, backup=None, agent=None`)

The `burpui.misc.backend.interface.BUIbackend.delete_backup()` function performs a backup deletion and returns an error message if the command failed.

**Parameters**

- **name** (`str`) – Client name
- **backup** (`int`) – Backup number
- **agent** (`str`) – What server to ask (only in multi-agent mode)

**Returns** An error message if the command failed

**delete\_client** (`client=None, keepconf=False, delcert=False, revoke=False, agent=None`)

The `burpui.misc.backend.interface.BUIbackend.delete_client()` function is used to delete a client from burp's configuration.

**Parameters**

- **client** (*str*) – The name of the client to remove
- **keepconf** (*bool*) – Whether to keep the conf (in order to just revoke/delete the certs for instance)
- **delcert** (*bool*) – Whether to delete the associated certificate
- **revoke** (*bool*) – Whether to revoke the associated certificate
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A list of notifications to return to the UI (success or failure)

**expand\_path** (*path=None, source=None, client=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.expand_path()` function is used to expand path of file inclusions glob the user can set in the setting panel. This function is also a *proxy* for multi-agent setup.

**Parameters**

- **path** (*str*) – The glob/path to expand
- **source** (*str*) – In which file are we working
- **client** (*str*) – The client name when working on client files
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A list of files or an empty list

**get\_all\_clients** (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_all_clients()` function returns a list containing all the clients with their states.

**Parameters** **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A list of clients

Example:

```
[
  {
    "last": "2015-10-02 08:20:03",
    "name": "client1",
    "state": "idle",
  },
  {
    "last": "2015-01-25 13:32:00",
    "name": "client2",
    "state": "idle"
  }
]
```

**get\_attr** (*name, default=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_attr()` function returns the given attribute or default.

**get\_backup\_logs** (*number, client, forward=False, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_backup_logs()` function is used to retrieve the burp logs depending the burp-server version.

**Parameters**

- **number** (*int*) – Backup number to work on
- **client** (*str*) – Client name to work on
- **forward** (*bool*) – Is the client name needed in later process
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** Dict containing the backup log

Example:

```
{
  "dir": {
    "changed": 0,
    "deleted": 0,
    "new": 17,
    "scanned": 30246,
    "total": 30246,
    "unchanged": 30229
  },
  "duration": 436,
  "efs": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "encrypted": false,
  "end": 1443767237,
  "files": {
    "changed": 47,
    "deleted": 2,
    "new": 2,
    "scanned": 227377,
    "total": 227377,
    "unchanged": 227328
  },
  "files_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "hardlink": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 28,
    "total": 28,
    "unchanged": 28
  },
  "meta": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
```

(continues on next page)

(continued from previous page)

```

        "scanned": 58,
        "total": 58,
        "unchanged": 58
    },
    "meta_enc": {
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 0,
        "total": 0,
        "unchanged": 0
    },
    "number": 576,
    "received": 11691704,
    "softlink": {
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 9909,
        "total": 9909,
        "unchanged": 9909
    },
    "special": {
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 1,
        "total": 1,
        "unchanged": 1
    },
    "start": 1443766801,
    "total": {
        "changed": 47,
        "deleted": 2,
        "new": 19,
        "scanned": 267619,
        "total": 267619,
        "unchanged": 267553
    },
    "totsize": 52047768383,
    "vssfooter": {
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 0,
        "total": 0,
        "unchanged": 0
    },
    "vssfooter_enc": {
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 0,
        "total": 0,
        "unchanged": 0
    },
    "vssheader": {

```

(continues on next page)

(continued from previous page)

```
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 0,
        "total": 0,
        "unchanged": 0
    },
    "vssheader_enc": {
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 0,
        "total": 0,
        "unchanged": 0
    },
    "windows": "false"
}
```

**get\_client** (*name=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_client()` function returns a list of dict representing the backups of a given client.

**Parameters**

- **name** (*str*) – Client name
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A list of backups

Example:

```
[
  {
    "date": "2015-01-25 13:32:00",
    "deletable": true,
    "encrypted": true,
    "number": "1",
    "received": 889818873,
    "size": 35612321050
  }
]
```

**get\_client\_filtered** (*name=None, limit=-1, page=None, start=None, end=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_client_filtered()` function returns a list of dict representing the backups of a given client filtered by the given criteria.

**Parameters**

- **name** (*str*) – Client name
- **limit** (*int*) – Number of element to return, -1 for not limit
- **page** (*int*) – What page to retrieve
- **start** (*int*) – Return elements after this date
- **end** (*int*) – Return elements until this date
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A list of backups

Example:

```
[
  {
    "date": "2015-01-25 13:32:00",
    "deletable": true,
    "encrypted": true,
    "number": "1",
    "received": 889818873,
    "size": 35612321050
  }
]
```

**get\_client\_labels** (*client=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_client_labels()` function returns a list of labels (if any) for a given client.

---

**Note:** Labels are only available since Burp 2.0.34

---

#### Parameters

- **client** (*str*) – The client for which you want the labels
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A list of labels or an empty list

**get\_client\_status** (*name=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_client_status()` function returns the status of a given client with its last stats.

#### Parameters

- **name** (*str*) – What client status do we want
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** The last status of a given client

Example:

```
{
  "state": "idle",
  "percent": null,
  "phase": null,
  "last": "never"
}
```

**get\_client\_version** (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_client_version()` function returns the client version used to connect to the server.

**Parameters** **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** Burp client version

**get\_clients\_report** (*clients, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_clients_report()` function returns the computed/compacted data to display clients report.

**Parameters**

- **clients** (*list*) – List of clients as returned by `burpui.misc.backend.interface.BUIbackend.get_all_clients()`
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A dict with the computed data

**get\_counters** (*name=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_counters()` function returns a dict of counters for a given client while it performs a backup.

**Parameters**

- **name** (*str*) – Name of the client for which you'd like stats
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A dict of counters

**get\_file** (*path, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_file()` function is used to retrieve a file on a remote agent.

**get\_parser** (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_parser()` function returns the parser of the current backend.

**get\_parser\_attr** (*attr=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_parser_attr()` function is used to retrieve some attributes from the Parser. This function is useful in multi-agent mode because the front-end needs to access the backend attributes through the agents.

**Parameters**

- **attr** (*str*) – Name of the attribute to retrieve
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** The requested attribute or an empty list

**get\_server\_version** (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_server_version()` function returns the server version (if any).

**Parameters** **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** Burp server version

**get\_tree** (*name=None, backup=None, root=None, level=-1, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_tree()` function returns a list of dict representing files/dir (with their attr) within a given path

**Parameters**

- **name** (*str*) – Client name
- **backup** (*int*) – Backup number
- **root** (*str*) – Root path to look into
- **level** (*int*) – Level of the tree relative to its root
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A list of files/dir within the given path with their attr



Example:

```
[
  {
    "date": "2015-01-23 20:00:07",
    "gid": "0",
    "inodes": "168",
    "mode": "drwxr-xr-x",
    "name": "/",
    "parent": "",
    "fullname": "/",
    "level": -1,
    "size": "12.0KiB",
    "type": "d",
    "uid": "0",
    "folder": True,
    "children": []
  }
]
```

**is\_backup\_running** (*name=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.is_backup_running()` functions tells you if a given client is currently performing a backup.

#### Parameters

- **name** (*str*) – Name of the client
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** True or False

**is\_one\_backup\_running** (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.is_one_backup_running()` function tells you if at least one backup is running.

**Parameters** **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A list of running clients

**is\_server\_backup** (*client=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.is_server_backup()` function is used to know if there is a server-initiated backup file in place.

#### Parameters

- **client** (*str*) – The name of the client to look for
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** True or False

**is\_server\_restore** (*client=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.is_server_restore()` function is used to know if there is a server-initiated restoration file in place and retrieve its content in order to edit it.

#### Parameters

- **client** (*str*) – The name of the client to look for
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A dict representing the content of the server-initiated restoration file

**read\_conf\_cli** (*client=None, conf=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.read_conf_cli()` function works the same way as the `burpui.misc.backend.interface.BUIbackend.read_conf_srv()` function but for the client config file.

**read\_conf\_srv** (*conf=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.read_conf_srv()` function returns a dict of options present in the server config file.

#### Parameters

- **conf** (*str*) – Complementary configuration file (for instance, file inclusions)
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** Dict of options

Example:

```
{
  "boolean": [
    {
      "name": "hardlinked_archive",
      "value": false
    },
    {
      "name": "syslog",
      "value": true
    }
  ],
  "clients": [
    {
      "name": "client1",
      "value": "/etc/burp/clientconffdir/client1"
    },
    {
      "name": "client2",
      "value": "/etc/burp/clientconffdir/client2"
    }
  ],
  "common": [
    {
      "name": "mode",
      "value": "server"
    },
    {
      "name": "directory",
      "value": "/srv/burp"
    }
  ],
  "includes": [],
  "includes_ext": [],
  "integer": [
    {
      "name": "port",
      "value": 4971
    },
    {
      "name": "status_port",
      "value": 4972
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    },
    {
        "name": "max_children",
        "value": 5
    },
    {
        "name": "max_status_children",
        "value": 5
    }
],
"multi": [
    {
        "name": "keep",
        "value": [
            "7",
            "4",
            "4"
        ]
    },
    {
        "name": "timer_arg",
        "value": [
            "12h",
            "Mon,Tue,Thu,Fri,17,18,19,20,21,22,23",
            "Wed,Sat,Sun,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,
↪22,23"
        ]
    }
]
}

```

**restore\_files** (*name=None, backup=None, files=None, strip=None, archive='zip', password=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.restore_files()` function performs a restoration and returns a tuple containing the path of the generated archive and/or a message if an error happened.

#### Parameters

- **name** (*str*) – Client name
- **backup** (*int*) – Backup number
- **files** (*str*) – A string representing a list of files to restore

Example:

```
['/etc/passwd', '/etc/shadow']
```

#### Parameters

- **strip** (*int*) – Number of parent directories to strip while restoring files
- **archive** (*str*) – Format of the generated archive (may be zip, tar.gz or tar.bz2)
- **password** (*str*) – Password for encrypted backups
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A tuple with the generated archive path and/or an error message

**revocation\_enabled** (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.revocation_enabled()` function is used to know if the revocation feature is enabled or not.

**Parameters** **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** True or False

**server\_backup** (*client=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.server_backup()` function is used to schedule a server-side initiated backup.

**Parameters**

- **client** (*str*) – Client name
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A list of notifications to return to the UI (success or failure)

**server\_restore** (*client=None, backup=None, files=None, strip=None, force=None, prefix=None, restoreto=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.server_restore()` function is used to schedule a server-side initiated restoration.

**Parameters**

- **client** (*str*) – Client name
- **backup** (*int*) – Backup number
- **files** (*str*) – List of files to restore
- **strip** (*int*) – Number of leading path to strip
- **force** (*bool*) – Whether to force overriding files or not
- **prefix** (*str*) – Where to restore files
- **restoreto** – Restore on an other client
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A list of notifications to return to the UI (success or failure)

**status** (*query='\n', timeout=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.status()` method is used to send queries to the Burp server

**Parameters**

- **query** (*str*) – Query to send to the server
- **timeout** (*int*) – Query timeout in seconds
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** The output returned by the server parsed as an array

Example:

```
[
  "client1      2      i      576 0 1443766803",
  "client2      2      i      1 0 1422189120"
]
```

**store\_conf\_cli** (*data*, *client=None*, *conf=None*, *template=False*, *agent=None*)

The `burpui.misc.backend.interface.BUIbackend.store_conf_cli()` function works the same way as the `burpui.misc.backend.interface.BUIbackend.store_conf_srv()` function but for the client config file. It takes an extra parameter:

**Parameters** **client** (*str*) – Name of the client for which to apply this config

**store\_conf\_srv** (*data*, *conf=None*, *agent=None*)

The `burpui.misc.backend.interface.BUIbackend.store_conf_srv()` functions is used to save the new settings in the configuration file.

**Parameters**

- **data** (*dict*) – Data as sent by the web-form
- **conf** (*str*) – Force the file path (for file inclusions for instance)
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A list of notifications to return to the UI (success or failure)

Example:

```
[[0, "Success"]]
```

**version** (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.version()` function returns the version of the given agent.

## 1.20.6 Parser

Here is the *parser* interface definition in order to implement a new parser.

**class** `burpui.misc.parser.interface.BUIparser`

`burpui.misc.parser.interface.BUIparser` defines a generic interface for burp configuration files parser.

**cancel\_backup** (*name=None*)

`burpui.misc.parser.interface.BUIparser.cancel_backup()` called by `burpui.misc.backend.interface.BUIbackend.cancel_server_backup()` in order to cancel a server-initiated backup.

**Parameters** **name** (*str*) – Client name

**Returns** A list of notifications to return to the UI (success or failure)

**cancel\_restore** (*name=None*)

`burpui.misc.parser.interface.BUIparser.cancel_restore()` called by `burpui.misc.backend.interface.BUIbackend.cancel_server_restore()` in order to cancel a server-initiated restoration.

**Parameters** **name** (*str*) – Client name

**Returns** A list of notifications to return to the UI (success or failure)

**is\_client\_revoked** (*client=None*)

`burpui.misc.parser.interface.BUIparser.is_client_revoked()` is used to check if a given client has its certificate revoked or not.

**Parameters** **client** (*str*) – The name of the client to check

**Returns** True or False

**list\_clients()**

*burpui.misc.parser.interface.BUIparser.list\_clients()* is used to retrieve a list of clients with their configuration file.

**Returns** A list of clients with their configuration file

**list\_templates()**

*burpui.misc.parser.interface.BUIparser.list\_templates()* is used to retrieve a list of templates with their absolute paths.

**Returns** A list of templates

**param(name, obj='server\_conf', client=None)**

*burpui.misc.parser.interface.BUIparser.param()* lookup for a given param in the conf.

**Parameters**

- **name** (*str*) – Param name
- **obj** (*str*) – Object to look param for
- **client** (*str*) – Search for a given client param

**Returns** The asked param

**path\_expander(pattern=None, source=None, client=None)**

*burpui.misc.parser.interface.BUIparser.path\_expander()* is used to expand path of file inclusions glob the user can set in the setting panel.

**Parameters**

- **pattern** (*str*) – The glob/path to expand
- **source** (*str*) – What file we are working in
- **client** (*str*) – The client name when working on client files

**Returns** A list of files or an empty list

**read\_backup(name=None)**

*burpui.misc.parser.interface.BUIparser.read\_backup()* called by *burpui.misc.backend.interface.BUIbackend.is\_server\_backup()* in order to test the existence of a server-initiated backup file.

**Parameters** **name** (*str*) – Client name

**Returns** A True if the file is found, else False.

**read\_client\_conf(client=None, conf=None, template=False)**

*burpui.misc.parser.interface.BUIparser.read\_client\_conf()* is called by *burpui.misc.backend.interface.BUIbackend.read\_conf\_cli()* in order to parse the burp-clients configuration files.

It works the same way as *burpui.misc.parser.interface.BUIparser.read\_server\_conf()*

**read\_restore(name=None)**

*burpui.misc.parser.interface.BUIparser.read\_restore()* called by *burpui.misc.backend.interface.BUIbackend.is\_server\_restore()* in order to read a server-initiated restoration file.

**Parameters** **name** (*str*) – Client name

**Returns** A dict describing the content of the file

**read\_server\_conf** (*conf=None*)

*burpui.misc.parser.interface.BUIparser.read\_server\_conf()* is called by *burpui.misc.backend.interface.BUIbackend.read\_conf\_srv()* in order to parse the burp-server configuration file.

**Parameters** **conf** (*str*) – Complementary configuration file (for instance, file inclusions)

**Returns** Dict of options

Example:

```
{
  "boolean": [
    {
      "name": "hardlinked_archive",
      "value": false
    },
    {
      "name": "syslog",
      "value": true
    },
  ],
  "clients": [
    {
      "name": "client1",
      "value": "/etc/burp/clientconffdir/client1"
    },
    {
      "name": "client2",
      "value": "/etc/burp/clientconffdir/client2"
    },
  ],
  "common": [
    {
      "name": "mode",
      "value": "server"
    },
    {
      "name": "directory",
      "value": "/srv/burp"
    },
  ],
  "includes": [],
  "includes_ext": [],
  "integer": [
    {
      "name": "port",
      "value": 4971
    },
    {
      "name": "status_port",
      "value": 4972
    },
    {
      "name": "max_children",
      "value": 5
    },
    {
      "name": "max_status_children",
```

(continues on next page)

(continued from previous page)

```

        "value": 5
    },
    ],
    "multi": [
        {
            "name": "keep",
            "value": [
                "7",
                "4",
                "4"
            ]
        },
        {
            "name": "timer_arg",
            "value": [
                "12h",
                "Mon,Tue,Thu,Fri,17,18,19,20,21,22,23",
                "Wed,Sat,Sun,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,
↪22,23"
            ]
        },
    ],
    ],
    "hierarchy": [
        {
            "children": [
                {
                    "children": [],
                    "dir": "/tmp/burp/conf.d",
                    "full": "/tmp/burp/conf.d/empty.conf",
                    "name": "empty.conf",
                    "parent": "/tmp/burp/burp-server.conf"
                },
                {
                    "children": [],
                    "dir": "/tmp/burp/conf.d",
                    "full": "/tmp/burp/conf.d/ipv4.conf",
                    "name": "ipv4.conf",
                    "parent": "/tmp/burp/burp-server.conf"
                }
            ],
            "dir": "/tmp/burp",
            "full": "/tmp/burp/burp-server.conf",
            "name": "burp-server.conf",
            "parent": null
        }
    ]
}

```

**remove\_client** (*client=None, keepconf=False, delcert=False, revoke=False, template=False*)

*burpui.misc.parser.interface.BUIparser.remove\_client()* is used to delete a client from burp's configuration.

#### Parameters

- **client** (*str*) – The name of the client to remove
- **keepconf** – Whether to keep the conf (in order to just revoke/delete the cert)
- **keepconf** – bool



- **delcert** (*bool*) – Whether to delete the associated certificate
- **revoke** (*bool*) – Whether to revoke the associated certificate
- **template** (*bool*) – Whether we remove a template

**Returns** A list of notifications to return to the UI (success or failure)

**remove\_conf** (*path=None*)

*burpui.misc.parser.interface.BUIparser.remove\_conf()* is used to remove a configuration file. It only works for “included” files within the server configuration file.

**Parameters** **path** (*str*) – The path of the file to remove

**server\_initiated\_backup** (*name=None*)

*burpui.misc.parser.interface.BUIparser.server\_initiated\_backup()* called by *burpui.misc.backend.interface.BUIbackend.server\_backup()* in order to create a server-initiated backup file.

**Parameters** **name** (*str*) – Client name

**Returns** A list of notifications to return to the UI (success or failure)

**server\_initiated\_restoration** (*name=None, backup=None, files=None, strip=None, force=None, prefix=None, restoreto=None*)

*burpui.misc.parser.interface.BUIparser.server\_initiated\_restoration()* called by *burpui.misc.backend.interface.BUIbackend.server\_restore()* in order to create server-initiated restoration file.

**Parameters**

- **name** (*str*) – Client name
- **backup** (*int*) – Backup number
- **files** (*str*) – List of files to restore
- **strip** (*int*) – Number of leading path to strip
- **force** (*bool*) – Whether to force overriding files or not
- **prefix** (*str*) – Where to restore files
- **agent** (*str*) – What server to ask (only in multi-agent mode)

**Returns** A list of notifications to return to the UI (success or failure)

**store\_client\_conf** (*data, client=None, conf=None, template=False*)

*burpui.misc.parser.interface.BUIparser.store\_client\_conf()* is used by *burpui.misc.backend.BUIbackend.store\_conf\_cli()*.

It works the same way as *burpui.misc.parser.interface.BUIparser.store\_conf()* with an extra parameter:

**Parameters**

- **client** (*str*) – Name of the client for which to apply this config
- **conf** (*str*) – The explicit filename of the conf
- **template** (*bool*) – Is this file a template

**store\_conf** (*data, conf=None, client=None, mode='srv', insecure=False, template=False*)

*burpui.misc.parser.interface.BUIparser.store\_conf()* is used to store the configuration from the web-ui into the actual configuration files. It is used by *burpui.misc.backend.BUIbackend.store\_conf\_srv()*.

**Parameters**

- **data** (*dict*) – Data sent by the web-form
- **conf** (*str*) – Force the file path (for file inclusions for instance)
- **client** (*str*) – Client name
- **mode** (*str*) – We actually use the same method for clients and server files
- **insecure** (*bool*) – Used for the CLI
- **template** (*bool*) – Is it a template

**Returns** A list of notifications to return to the UI (success or failure)

Example:

```
[[0, "Success"]]
```

### 1.20.7 Auth

Here is the *auth* interface definition in order to implement a new authentication backend. It is composed by two classes.

**class** `burpui.misc.auth.interface.BUIhandler`

The `burpui.misc.auth.interface.BUIhandler` class maintains a list of Burp-UI users.

**Parameters** `app` (`burpui.server.BUIServer`) – Instance of the app we are running in

**remove** (*name*)

The `burpui.misc.auth.interface.BUIhandler.remove()` function allows to remove a user from the cache.

**Parameters** `name` (*str*) – Name of the user to remove

**user** (*name=None, refresh=False*)

The `burpui.misc.auth.interface.BUIhandler.user()` function returns the `flask_login:flask_login.UserMixin` object corresponding to the given user name.

**Parameters**

- **name** (*str*) – Name of the user
- **refresh** (*bool*) – Whether we need to re-create a fresh user or not

**Returns** `burpui.misc.auth.interface.BUIuser`

**class** `burpui.misc.auth.interface.BUIuser`

The `burpui.misc.auth.interface.BUIuser` class extends the `flask_login:flask_login.UserMixin` class.

**is\_active**

**Returns** True if user is active, otherwise False

**is\_admin**

If no ACL engine is loaded, every logged-in user will be granted admin rights :returns: True if the user is admin, otherwise False

**is\_authenticated**

**Returns** True if a user is authenticated, otherwise False

**is\_moderator**

If no ACL engine is loaded, every logged-in user will be granted moderator rights :returns: True if the user is moderator, otherwise False

**login** (*passwd=None*)

The `burpui.misc.auth.interface.BUIuser.login()` function checks if the provided user-name and password match.

**Parameters** `passwd` (*str*) – Password

**Returns** True if the name and password match, otherwise False

## 1.20.8 ACL

Here is the *acl* interface definition in order to implement a new acl backend. It is composed by two classes.

**class** `burpui.misc.acl.interface.BUIaclLoader` (*app=None*)

The `burpui.misc.acl.interface.BUIaclLoader` class is used to load the actual ACL backend

**acl**

Property to retrieve the backend

**grants**

Property to retrieve the list of grants

**groups**

Property to retrieve the list of groups with their members

**init\_app** (*app*)

Register the given app

**reload** ()

Reload the backend

**class** `burpui.misc.acl.interface.BUIacl` (*app=None*)

The `burpui.misc.acl.interface.BUIacl` class represents the ACL engine.

**clients** (*username=None, server=None*)

`burpui.misc.acl.interface.BUIacl.clients()` returns a list of allowed clients for a given user.

**Parameters**

- **username** (*str*) – Username to check
- **server** (*str*) – Server name. Used in multi-agent mode

**Returns** A list of clients

**Return type** list

Deprecated since version 0.6.0.

**init\_app** (*app*)

Register the given app

**is\_admin** (*username=None*)

`burpui.misc.acl.interface.BUIacl.is_admin()` is used to know if a user has administrator rights.

**Parameters** `username` (*str*) – Username to check

**Returns** True if the user has admin rights, otherwise False

**Return type** bool

**is\_client\_allowed** (*username=None, client=None, server=None*)

*burpui.misc.acl.interface.BUIacl.is\_client\_allowed()* tells us if a given user has access to a given client on a given server.

**Parameters**

- **username** (*str*) – Username to check
- **client** (*str*) – Client to check
- **server** (*str*) – Server to check

**Returns** True if username is granted, otherwise False

**Return type** bool

**is\_client\_rw** (*username=None, client=None, server=None*)

*burpui.misc.acl.interface.BUIacl.is\_client\_rw()* tells us if a given user has access to a given client on a given server in RW mode.

**Parameters**

- **username** (*str*) – Username to check
- **client** (*str*) – Client to check
- **server** (*str*) – Server to check

**Returns** True if username is granted, otherwise False

**Return type** bool

**is\_moderator** (*username=None*)

*burpui.misc.acl.interface.BUIacl.is\_moderator()* is used to know if a user has moderator rights.

**Parameters** **username** (*str*) – Username to check

**Returns** True if the user has moderator rights, otherwise False

**Return type** bool

**is\_server\_allowed** (*username=None, server=None*)

*burpui.misc.acl.interface.BUIacl.is\_server\_allowed()* tells us if a given user has access to a given server.

**Parameters**

- **username** (*str*) – Username to check
- **server** (*str*) – Server to check

**Returns** True if username is granted, otherwise False

**Return type** bool

**is\_server\_rw** (*username=None, server=None*)

*burpui.misc.acl.interface.BUIacl.is\_server\_rw()* tells us if a given user has access to a given server in RW mode.

**Parameters**

- **username** (*str*) – Username to check
- **server** (*str*) – Server to check

**Returns** True if username is granted, otherwise False

**Return type** bool

**servers** (*username=None*)

*burpui.misc.acl.interface.BUIacl.servers()* returns a list of allowed servers for a given user.

**Parameters** **username** (*str*) – Username to check

**Returns** A list of servers

**Return type** list

Deprecated since version 0.6.0.

**class** *burpui.misc.acl.meta.BUIgrantHandler* (*app=None*)

This class is here to handle grants in a generic way. It will automatically merge grants from various backends that register to it

**changed** (*sid*)

detect a configuration change

**clients** (*username=None, server=None*)

*burpui.misc.acl.interface.BUIacl.clients()* returns a list of allowed clients for a given user.

**Parameters**

- **username** (*str*) – Username to check
- **server** (*str*) – Server name. Used in multi-agent mode

**Returns** A list of clients

**Return type** list

Deprecated since version 0.6.0.

**grants**

grants managed by our handler

**groups**

groups managed by our handler

**id**

current handler id, used to detect configuration changes

**init\_app** (*app*)

Register the given app

**is\_admin** (*username*)

See *burpui.misc.acl.interface.BUIacl.is\_admin()*

**is\_client\_allowed** (*username=None, client=None, server=None*)

See *burpui.misc.acl.interface.BUIacl.is\_client\_allowed()*

**is\_client\_rw** (*username=None, client=None, server=None*)

See *burpui.misc.acl.interface.BUIacl.is\_client\_rw()*

**is\_moderator** (*username*)

See *burpui.misc.acl.interface.BUIacl.is\_moderator()*

**is\_server\_allowed** (*username=None, server=None*)

See *burpui.misc.acl.interface.BUIacl.is\_server\_allowed()*

**is\_server\_rw** (*username=None, server=None*)

See `burpui.misc.acl.interface.BUIacl.is_server_rw()`

**opt** (*key, default=False*)

access a given option

**options**

options of our ACL engine

**register\_backend** (*name, backend*)

register a new ACL backend

#### Parameters

- **name** (*str*) – Backend name
- **backend** (`burpui.misc.acl.interface.BUIaclLoader`) – ACL Backend

**reset** (*reset\_from*)

a configuration change occurred, we reload our grants and groups

**servers** (*username=None*)

`burpui.misc.acl.interface.BUIacl.servers()` returns a list of allowed servers for a given user.

**Parameters** **username** (*str*) – Username to check

**Returns** A list of servers

**Return type** list

Deprecated since version 0.6.0.

**set\_admin** (*admins*)

parse and set the admins

**set\_grant** (*name, grant*)

parse and set the given grants

**set\_group** (*name, members*)

parse and set the given group

**set\_moderator** (*moderators*)

parse and set the moderators

**set\_moderator\_grants** (*grants*)

parse and set the moderators grants

## 1.20.9 Plugins

Since *v0.6.0*, you can write your own external plugins. For now, only *authentication* and *acl* plugins are supported.

### Authentication

You will find here a fully working example of an external *authentication* plugin. Please refer to the [Auth API](#) page for more details.

```
1 from burpui.misc.auth import interface
2
3 __type__ = 'auth'
4
```

(continues on next page)

(continued from previous page)

```

5 class UserHandler(interface.BUIhandler):
6     name = 'CUSTOM'
7     priority = 1000
8
9     def __init__(self, app):
10         self.users = {
11             'toto': CustomUser('toto', 'toto'),
12             'tata': CustomUser('tata', 'tata'),
13             'titi': CustomUser('titi', 'titi'),
14             'tutu': CustomUser('tutu', 'tutu'),
15         }
16
17     def user(self, name):
18         return self.users.get(name, None)
19
20     @property
21     def loader(self):
22         return self
23
24 class CustomUser(interface.BUIuser):
25     def __init__(self, name, password):
26         self.name = self.id = name
27         self.password = password
28
29     def login(self, passwd):
30         self.authenticated = passwd == self.password
31         return self.authenticated

```

Line 1 is mandatory since you must implement the *auth* interface in order for your plugin to work.

Line 3 `__type__ = 'auth'` defines a *auth* plugin.

Line 6 defines your *auth* backend name.

The rest of the code is just a minimal implementation of the *auth* interface.

This plugin defines four hardcoded users: *toto*, *tata*, *titi*, *tutu* with respectively the same passwords as their username.

You can put this code in a file called *custom.py*, save this file in */etc/burp/plugins* for instance, and set `plugins = /etc/burp/plugins`. The plugin will be automatically loaded.

---

**Note:** This is just an example, do not run this particular plugin in production!

---

## ACL

You will find here a fully working example of an external *acl* plugin. Please refer to the [ACL API](#) page for more details.

```

1 from burpui.misc.acl import interface
2
3 __type__ = 'acl'
4
5 class ACLloader(interface.BUIaclLoader):
6     name = 'CUSTOM:ACL'
7     priority = 1000

```

(continues on next page)

(continued from previous page)

```

8
9     def __init__(self, app):
10         self.app = app
11         self.admin = 'toto'
12         self._acl = CustomACL(self)
13
14     @property
15     def acl(self):
16         return self._acl
17
18     @property
19     def grants(self):
20         return None
21
22     @property
23     def groups(self):
24         return None
25
26     def reload(self):
27         """This method is used to reload the rules in case of config
28         change for instance"""
29         pass
30
31
32     class CustomACL(interface.BUIacl):
33
34         def __init__(self, loader):
35             self.loader = loader
36
37         def is_admin(self, username=None):
38             if not username:
39                 return False
40             return username == self.loader.admin
41
42         def is_moderator(self, username=None):
43             if not username:
44                 return False
45             return username == self.loader.admin
46
47         def is_client_rw(self, username=None, client=None, server=None):
48             if not username:
49                 return False
50             return username == self.loader.admin
51
52         def is_client_allowed(self, username=None, client=None, server=None):
53             if not username:
54                 return False
55             return username == self.loader.admin
56
57         def is_server_rw(self, username=None, server=None):
58             if not username:
59                 return False
60             return username == self.loader.admin
61
62         def is_server_allowed(self, username=None, server=None):
63             if not username:
64                 return False

```

(continues on next page)



(continued from previous page)

```
65         return username == self.loader.admin
```

Line 1 is mandatory since you must implement the *acl* interface in order for your plugin to work.

Line 3 `__type__ = 'acl'` defines a *acl* plugin.

Line 6 defines your *acl* backend name.

The rest of the code is just a minimal implementation of the *acl* interface.

This plugin defines a hardcoded admin user: *toto* which will be granted admin rights through the whole application.

You can put this code in a file called *custom\_acl.py*, save this file in */etc/burp/plugins* for instance, and set `plugins = /etc/burp/plugins`. The plugin will be automatically loaded.

---

**Note:** This is just an example, do not run this particular plugin in production!

---

ACL engine has built-in Groups support, to take full advantage of this feature, it is recommended to use the `meta_grants` object as shown below:

---

**Note:** The grant syntax is explained in the [ACL](#) documentation

---

```
1  from burpui.misc.acl.meta import meta_grants
2  from burpui.misc.acl import interface
3
4  from six import iteritems
5
6  __type__ = 'acl'
7
8  class ACLloader(interface.BUIaclLoader):
9      name = 'CUSTOM2:ACL'
10     priority = 1001
11
12     _groups = {
13         'gp1': {
14             'grants': '["server1", "server2"]', # this needs to be a string
15             'members': ['user1'],
16         },
17     }
18
19     def __init__(self, app):
20         self.app = app
21         self.admin = 'toto'
22         self.init_rules()
23         self._acl = meta_grants
24         # We need to register our backend in order to be notified of
25         # configuration changes in other registered backends.
26         # This will then call our 'reload' function in order to re-apply
27         # our grants.
28         meta_grants.register_backend(self.name, self)
29
30     def init_rules(self):
31         for gname, content in iteritems(self._groups):
32             meta_grants.set_group(gname, content['members'])
33             meta_grants.set_grant(gname, content['grants'])
```

(continues on next page)

(continued from previous page)

```
34
35     @property
36     def acl(self):
37         return self._acl
38
39     @property
40     def grants(self):
41         return self.acl.grants
42
43     @property
44     def groups(self):
45         return self._groups
46
47     def reload(self):
48         """This method is used to reload the rules in case of config
49         change for instance"""
50         self.init_rules()
```

You can omit either the `meta_grants.set_grant` or the `meta_grants.set_group` part if you like. For instance to define the grants of a given group using another ACL backend, and using your plugin to manage groups membership only.

---

## HTTP Routing Table

---

### /api

	77
GET /api/admin/acl/(backend)/admin, 66	GET /api/admin/acl/moderator, 63
GET /api/admin/acl/(backend)/admin/(member), 66	GET /api/admin/acl/moderators, 63
GET /api/admin/acl/(backend)/admins, 64	GET /api/admin/auth/(backend)/users, 63
GET /api/admin/acl/(backend)/grants, 65	GET /api/admin/auth/(backend)/users/(name), 63
GET /api/admin/acl/(backend)/grants/(name), 65	GET /api/admin/auth/backend/(backend), 77
GET /api/admin/acl/(backend)/group/(name), 66	GET /api/admin/auth/backends, 62
GET /api/admin/acl/(backend)/group/(name)/(member), 66	GET /api/admin/auth/users, 63
GET /api/admin/acl/(backend)/groups, 64	GET /api/admin/auth/users/(name), 63
GET /api/admin/acl/(backend)/groups/(name), 64	GET /api/admin/me, 75
GET /api/admin/acl/(backend)/isAdmin/(member), 77	GET /api/admin/me/session, 66
GET /api/admin/acl/(backend)/isModerator/(member), 77	GET /api/admin/me/session/(uuid:id), 66
GET /api/admin/acl/(backend)/isModerator/(member), 77	GET /api/admin/session/(user), 85
GET /api/admin/acl/(backend)/moderator, 63	GET /api/admin/session/(user)/(uuid:id), 85
GET /api/admin/acl/(backend)/moderator/(member), 63	GET /api/backup/(server)/server-backup/(name), 79
GET /api/admin/acl/(backend)/moderators, 63	GET /api/backup/server-backup/(name), 79
GET /api/admin/acl/admin, 66	GET /api/client/(server)/browse/(name)/(int:backup), 81
GET /api/admin/acl/admins, 64	GET /api/client/(server)/browseall/(name)/(int:backup), 80
GET /api/admin/acl/backend/(backend), 77	GET /api/client/(server)/labels/(name), 84
GET /api/admin/acl/backends, 64	GET /api/client/(server)/report/(name), 81
GET /api/admin/acl/grants, 65	GET /api/client/(server)/report/(name)/(int:backup), 81
GET /api/admin/acl/grants/(name), 65	GET /api/client/(server)/running, 74
GET /api/admin/acl/group, 66	GET /api/client/(server)/running/(name), 74
GET /api/admin/acl/groups, 64	GET /api/client/(server)/stats/(name), 85
GET /api/admin/acl/groups/(name), 64	GET /api/client/browse/(name)/(int:backup), 81
GET /api/admin/acl/groupsOf/(member), 77	GET /api/client/browseall/(name)/(int:backup), 80
GET /api/admin/acl/isAdmin/(member), 77	
GET /api/admin/acl/isModerator/(member), 77	

GET /api/client/labels/ (name), 84	GET /api/settings/ (server) /server-config, 67
GET /api/client/report/ (name), 81	GET /api/settings/ (server) /server-config/ (path:conf), 67
GET /api/client/report/ (name) / (int:backup), 81	GET /api/settings/ (server) /templates, 70
GET /api/client/running, 75	GET /api/settings/clients, 71
GET /api/client/running/ (name), 74	GET /api/settings/config/ (client), 78
GET /api/client/stats/ (name), 85	GET /api/settings/config/ (client) / (path:conf), 78
GET /api/clients/ (server) /all, 74	GET /api/settings/options, 71
GET /api/clients/ (server) /backup-running, 71	GET /api/settings/path-expander, 70
GET /api/clients/ (server) /report, 72	GET /api/settings/path-expander/ (client), 70
GET /api/clients/ (server) /running, 71	GET /api/settings/server-config, 67
GET /api/clients/ (server) /running/ (client), 71	GET /api/settings/server-config/ (path:conf), 67
GET /api/clients/ (server) /stats, 73	GET /api/settings/templates, 70
GET /api/clients/all, 74	GET /api/swagger.json, 77
GET /api/clients/backup-running, 71	POST /api/admin/acl/ (backend) /grants, 65
GET /api/clients/report, 72	POST /api/admin/acl/ (backend) /grants/ (name), 65
GET /api/clients/running, 71	POST /api/admin/acl/ (backend) /groups, 65
GET /api/clients/running/ (client), 71	POST /api/admin/acl/ (backend) /groups/ (name), 65
GET /api/clients/stats, 73	POST /api/admin/acl/ (backend) /moderator, 64
GET /api/doc, 77	POST /api/admin/acl/ (backend) /moderator/ (member), 64
GET /api/misc/ (server) /about, 77	POST /api/admin/acl/grants, 65
GET /api/misc/ (server) /counters, 75	POST /api/admin/acl/grants/ (name), 65
GET /api/misc/ (server) /counters/ (client), 75	POST /api/admin/acl/groups, 65
GET /api/misc/ (server) /history, 76	POST /api/admin/acl/groups/ (name), 65
GET /api/misc/ (server) /history/ (client), 76	POST /api/admin/acl/moderator, 64
GET /api/misc/ (server) /monitor, 76	POST /api/admin/auth/ (backend) /users, 63
GET /api/misc/about, 77	POST /api/admin/auth/ (backend) /users/ (name), 63
GET /api/misc/counters, 75	POST /api/admin/auth/users, 63
GET /api/misc/counters/ (client), 75	POST /api/admin/auth/users/ (name), 63
GET /api/misc/history, 76	POST /api/misc/alert, 77
GET /api/misc/history/ (client), 76	POST /api/preferences/ui, 67
GET /api/misc/languages, 75	POST /api/restore/ (server) /archive/ (name) / (int:backup), 79
GET /api/misc/monitor, 76	POST /api/restore/archive/ (name) / (int:backup), 79
GET /api/preferences/ui, 67	POST /api/settings/ (server) /config/ (client), 78
GET /api/restore/ (server) /server-restore/ (name), 78	POST /api/settings/ (server) /config/ (client) / (path:conf), 78
GET /api/restore/server-restore/ (name), 78	POST /api/settings/ (server) /options, 71
GET /api/servers/report, 72	GET /api/settings/ (server) /path-expander, 70
GET /api/servers/stats, 74	POST /api/settings/ (server) /path-expander/ (client), 70
GET /api/settings/ (server) /clients, 71	POST /api/settings/ (server) /server-config, 67
GET /api/settings/ (server) /config/ (client), 78	
GET /api/settings/ (server) /config/ (client) / (path:conf), 78	
GET /api/settings/ (server) /options, 71	
GET /api/settings/ (server) /path-expander, 70	
GET /api/settings/ (server) /path-expander/ (client), 70	

POST /api/settings/(server)/server-config,70	DELETE /api/admin/acl/(backend)/grants,65
POST /api/settings/config/(client),78	DELETE /api/admin/acl/(backend)/grants/(name),65
POST /api/settings/config/(client)/(path:conf),78	DELETE /api/admin/acl/(backend)/group/(name),66
POST /api/settings/server-config,70	DELETE /api/admin/acl/(backend)/group/(name)/(member),66
POST /api/settings/server-config/(path:conf),70	DELETE /api/admin/acl/(backend)/groups,64
PUT /api/admin/acl/(backend)/admin,66	DELETE /api/admin/acl/(backend)/groups/(name),64
PUT /api/admin/acl/(backend)/admin/(member),66	DELETE /api/admin/acl/(backend)/moderator,63
PUT /api/admin/acl/(backend)/grants,65	DELETE /api/admin/acl/(backend)/moderator/(member),63
PUT /api/admin/acl/(backend)/grants/(name),65	DELETE /api/admin/acl/admin,66
PUT /api/admin/acl/(backend)/group/(name),66	DELETE /api/admin/acl/grants,65
PUT /api/admin/acl/(backend)/group/(name)/(member),66	DELETE /api/admin/acl/grants/(name),65
PUT /api/admin/acl/(backend)/groups,65	DELETE /api/admin/acl/group,66
PUT /api/admin/acl/(backend)/groups/(name),64	DELETE /api/admin/acl/groups,64
PUT /api/admin/acl/(backend)/moderator,64	DELETE /api/admin/acl/groups/(name),64
PUT /api/admin/acl/(backend)/moderator/(member),64	DELETE /api/admin/acl/moderator,63
PUT /api/admin/acl/(backend)/moderator/(member)/(member),64	DELETE /api/admin/auth/(backend)/users,63
PUT /api/admin/acl/admin,66	DELETE /api/admin/auth/(backend)/users/(name),62
PUT /api/admin/acl/grants,65	DELETE /api/admin/auth/users,63
PUT /api/admin/acl/grants/(name),65	DELETE /api/admin/auth/users/(name),63
PUT /api/admin/acl/group,66	DELETE /api/admin/me/session,67
PUT /api/admin/acl/groups,65	DELETE /api/admin/me/session/(uuid:id),67
PUT /api/admin/acl/groups/(name),65	DELETE /api/backup/(server)/server-backup/(name),80
PUT /api/admin/acl/moderator,64	DELETE /api/backup/server-backup/(name),80
PUT /api/admin/auth/(backend)/users,63	DELETE /api/client/(server)/report/(name),84
PUT /api/admin/auth/(backend)/users/(name),63	DELETE /api/client/(server)/report/(name)/(int:backup),84
PUT /api/backup/(server)/do-server-backup/(name),79	DELETE /api/client/report/(name),84
PUT /api/backup/do-server-backup/(name),79	DELETE /api/client/report/(name)/(int:backup),84
PUT /api/preferences/ui,67	DELETE /api/preferences/ui,67
PUT /api/restore/(server)/server-restore/(name),78	DELETE /api/restore/(server)/server-restore/(name),79
PUT /api/restore/server-restore/(name)/(int:backup),78	DELETE /api/restore/server-restore/(name),79
PUT /api/settings/(server)/config,71	DELETE /api/settings/(server)/config/(client),78
PUT /api/settings/(server)/template,70	DELETE /api/settings/(server)/config/(client)/(path:conf),78
PUT /api/settings/config,71	DELETE /api/settings/(server)/server-config,70
PUT /api/settings/template,71	
DELETE /api/admin/acl/(backend)/admin,65	
DELETE /api/admin/acl/(backend)/admin/(member),65	

```
DELETE /api/settings/(server)/server-config/(path:conf),  
70  
DELETE /api/settings/config/(client),  
78  
DELETE /api/settings/config/(client)/(path:conf),  
78  
DELETE /api/settings/server-config,70  
DELETE /api/settings/server-config/(path:conf),  
70
```

## A

acl (burpui.misc.acl.interface.BUIaclLoader attribute), 103

## B

BUIacl (class in burpui.misc.acl.interface), 103  
 BUIaclLoader (class in burpui.misc.acl.interface), 103  
 BUIbackend (class in burpui.misc.backend.interface), 85  
 BUIgrantHandler (class in burpui.misc.acl.meta), 105  
 BUIhandler (class in burpui.misc.auth.interface), 102  
 BUIparser (class in burpui.misc.parser.interface), 97  
 BUIuser (class in burpui.misc.auth.interface), 102

## C

cancel\_backup() (burpui.misc.parser.interface.BUIparser method), 97  
 cancel\_restore() (burpui.misc.parser.interface.BUIparser method), 97  
 cancel\_server\_backup() (burpui.misc.backend.interface.BUIbackend method), 86  
 cancel\_server\_restore() (burpui.misc.backend.interface.BUIbackend method), 86  
 changed() (burpui.misc.acl.meta.BUIgrantHandler method), 105  
 clients() (burpui.misc.acl.interface.BUIacl method), 103  
 clients() (burpui.misc.acl.meta.BUIgrantHandler method), 105  
 clients\_list() (burpui.misc.backend.interface.BUIbackend method), 86  
 create\_app() (in module burpui), 62

## D

del\_file() (burpui.misc.backend.interface.BUIbackend method), 86  
 delete\_backup() (burpui.misc.backend.interface.BUIbackend method), 86

delete\_client() (burpui.misc.backend.interface.BUIbackend method), 86

## E

expand\_path() (burpui.misc.backend.interface.BUIbackend method), 87

## G

get\_all\_clients() (burpui.misc.backend.interface.BUIbackend method), 87  
 get\_attr() (burpui.misc.backend.interface.BUIbackend method), 87  
 get\_backup\_logs() (burpui.misc.backend.interface.BUIbackend method), 87  
 get\_client() (burpui.misc.backend.interface.BUIbackend method), 90  
 get\_client\_filtered() (burpui.misc.backend.interface.BUIbackend method), 90  
 get\_client\_labels() (burpui.misc.backend.interface.BUIbackend method), 91  
 get\_client\_status() (burpui.misc.backend.interface.BUIbackend method), 91  
 get\_client\_version() (burpui.misc.backend.interface.BUIbackend method), 91  
 get\_clients\_report() (burpui.misc.backend.interface.BUIbackend method), 91  
 get\_counters() (burpui.misc.backend.interface.BUIbackend method), 92  
 get\_file() (burpui.misc.backend.interface.BUIbackend method), 92  
 get\_parser() (burpui.misc.backend.interface.BUIbackend method), 92  
 get\_parser\_attr() (burpui.misc.backend.interface.BUIbackend method), 92

`get_server_version()` (burpui.misc.backend.interface.BUIbackend method), 92

`get_tree()` (burpui.misc.backend.interface.BUIbackend method), 92

`grants` (burpui.misc.acl.interface.BUIaclLoader attribute), 103

`grants` (burpui.misc.acl.meta.BUIgrantHandler attribute), 105

`groups` (burpui.misc.acl.interface.BUIaclLoader attribute), 103

`groups` (burpui.misc.acl.meta.BUIgrantHandler attribute), 105

**I**

`id` (burpui.misc.acl.meta.BUIgrantHandler attribute), 105

`init_app()` (burpui.misc.acl.interface.BUIacl method), 103

`init_app()` (burpui.misc.acl.interface.BUIaclLoader method), 103

`init_app()` (burpui.misc.acl.meta.BUIgrantHandler method), 105

`is_active` (burpui.misc.auth.interface.BUIuser attribute), 102

`is_admin` (burpui.misc.auth.interface.BUIuser attribute), 102

`is_admin()` (burpui.misc.acl.interface.BUIacl method), 103

`is_admin()` (burpui.misc.acl.meta.BUIgrantHandler method), 105

`is_authenticated` (burpui.misc.auth.interface.BUIuser attribute), 102

`is_backup_running()` (burpui.misc.backend.interface.BUIbackend method), 93

`is_client_allowed()` (burpui.misc.acl.interface.BUIacl method), 104

`is_client_allowed()` (burpui.misc.acl.meta.BUIgrantHandler method), 105

`is_client_revoked()` (burpui.misc.parser.interface.BUIparser method), 97

`is_client_rw()` (burpui.misc.acl.interface.BUIacl method), 104

`is_client_rw()` (burpui.misc.acl.meta.BUIgrantHandler method), 105

`is_moderator` (burpui.misc.auth.interface.BUIuser attribute), 102

`is_moderator()` (burpui.misc.acl.interface.BUIacl method), 104

`is_moderator()` (burpui.misc.acl.meta.BUIgrantHandler method), 105

`is_one_backup_running()` (burpui.misc.backend.interface.BUIbackend method), 93

`is_server_allowed()` (burpui.misc.acl.interface.BUIacl method), 104

`is_server_allowed()` (burpui.misc.acl.meta.BUIgrantHandler method), 105

`is_server_backup()` (burpui.misc.backend.interface.BUIbackend method), 93

`is_server_restore()` (burpui.misc.backend.interface.BUIbackend method), 93

`is_server_rw()` (burpui.misc.acl.interface.BUIacl method), 104

`is_server_rw()` (burpui.misc.acl.meta.BUIgrantHandler method), 105

**L**

`list_clients()` (burpui.misc.parser.interface.BUIparser method), 97

`list_templates()` (burpui.misc.parser.interface.BUIparser method), 98

`login()` (burpui.misc.auth.interface.BUIuser method), 103

**O**

`opt()` (burpui.misc.acl.meta.BUIgrantHandler method), 106

`options` (burpui.misc.acl.meta.BUIgrantHandler attribute), 106

**P**

`param()` (burpui.misc.parser.interface.BUIparser method), 98

`path_expander()` (burpui.misc.parser.interface.BUIparser method), 98

**R**

`read_backup()` (burpui.misc.parser.interface.BUIparser method), 98

`read_client_conf()` (burpui.misc.parser.interface.BUIparser method), 98

`read_conf_cli()` (burpui.misc.backend.interface.BUIbackend method), 93

`read_conf_srv()` (burpui.misc.backend.interface.BUIbackend method), 94

`read_restore()` (burpui.misc.parser.interface.BUIparser method), 98

`read_server_conf()` (burpui.misc.parser.interface.BUIparser method), 98

`register_backend()` (burpui.misc.acl.meta.BUIgrantHandler method), 106



[reload\(\)](#) (burpui.misc.acl.interface.BUIaclLoader method), [103](#)  
[remove\(\)](#) (burpui.misc.auth.interface.BUIhandler method), [102](#)  
[remove\\_client\(\)](#) (burpui.misc.parser.interface.BUIparser method), [100](#)  
[remove\\_conf\(\)](#) (burpui.misc.parser.interface.BUIparser method), [101](#)  
[reset\(\)](#) (burpui.misc.acl.meta.BUIgrantHandler method), [106](#)  
[restore\\_files\(\)](#) (burpui.misc.backend.interface.BUIbackend method), [95](#)  
[revocation\\_enabled\(\)](#) (burpui.misc.backend.interface.BUIbackend method), [96](#)

## S

[server\\_backup\(\)](#) (burpui.misc.backend.interface.BUIbackend method), [96](#)  
[server\\_initiated\\_backup\(\)](#) (burpui.misc.parser.interface.BUIparser method), [101](#)  
[server\\_initiated\\_restoration\(\)](#) (burpui.misc.parser.interface.BUIparser method), [101](#)  
[server\\_restore\(\)](#) (burpui.misc.backend.interface.BUIbackend method), [96](#)  
[servers\(\)](#) (burpui.misc.acl.interface.BUIacl method), [105](#)  
[servers\(\)](#) (burpui.misc.acl.meta.BUIgrantHandler method), [106](#)  
[set\\_admin\(\)](#) (burpui.misc.acl.meta.BUIgrantHandler method), [106](#)  
[set\\_grant\(\)](#) (burpui.misc.acl.meta.BUIgrantHandler method), [106](#)  
[set\\_group\(\)](#) (burpui.misc.acl.meta.BUIgrantHandler method), [106](#)  
[set\\_moderator\(\)](#) (burpui.misc.acl.meta.BUIgrantHandler method), [106](#)  
[set\\_moderator\\_grants\(\)](#) (burpui.misc.acl.meta.BUIgrantHandler method), [106](#)  
[status\(\)](#) (burpui.misc.backend.interface.BUIbackend method), [96](#)  
[store\\_client\\_conf\(\)](#) (burpui.misc.parser.interface.BUIparser method), [101](#)  
[store\\_conf\(\)](#) (burpui.misc.parser.interface.BUIparser method), [101](#)  
[store\\_conf\\_cli\(\)](#) (burpui.misc.backend.interface.BUIbackend method), [96](#)  
[store\\_conf\\_srv\(\)](#) (burpui.misc.backend.interface.BUIbackend method), [97](#)

## U

[user\(\)](#) (burpui.misc.auth.interface.BUIhandler method), [102](#)

## V

[version\(\)](#) (burpui.misc.backend.interface.BUIbackend method), [97](#)