# MONARC Stats Service

***Release 0.3.0***

**CASES Luxembourg**

**Feb 25, 2021**

## Contents

This component (https://github.com/monarc-project/stats-service) provides an API in order to **collect** statistics from one or several MONARC (https://github.com/monarc-project/MonarcAppFO) instances and to **return** these statistics with different filters and aggregation methods.

It can be deployed just next to a MONARC instance or on a dedicated server.

The collected statistics can be sent to an other stats instance.

# 1 Technical considerations

## 1.1 Installation

### 1.1.1 Prerequisites

Generally speaking, requirements are the following:

- A GNU/Linux distribution (tested on Debian and Ubuntu);
- Python: version >= 3.6.12 (preferably use pyenv (https://github.com/pyenv/pyenv)) and a dependency manager (for example Poetry (https://python-poetry.org));
- A PostgreSQL server 12.x: persistent storage.

Additionally:

- A cron daemon: running scheduled tasks for pushing or pulling stats data.

Creation of a PostgreSQL user:

```
$ sudo apt install postgresql
$ sudo -u postgres createuser <username>
$ sudo -u postgres psql
psql (11.2 (Ubuntu 11.2-1))
Type "help" for help.
postgres=# ALTER USER <username> WITH encrypted password '<password>';
postgres=# ALTER USER <username> WITH SUPERUSER;
ALTER ROLE
postgres-# \q
```

The user name and password chosen must be specified later in the configuration file.

### 1.1.2 Deployment

The service can be deployed via several ways:

### From the source

```
$ sudo apt install python3-pip python3-venv
$ curl -sSL https://raw.githubusercontent.com/python-poetry/poetry/master/get-poetry.
↪py | python3
$ echo  'export PATH="$PATH:$HOME/.poetry/bin"' >> ~/.bashrc
$ . ~/.bashrc

$ git clone https://github.com/monarc-project/stats-service
$ cd stats-service/
$ npm install
$ cp instance/production.py.cfg instance/production.py  # configure appropriately
$ poetry install # install the application
$ export STATS_CONFIG=production.py
$ FLASK_APP=runserver.py poetry run flask db_create # database creation
$ FLASK_APP=runserver.py poetry run flask db_init # database initialization

$ FLASK_APP=runserver.py FLASK_ENV=development poetry run flask run
```

For production you should use Gunicorn (https://gunicorn.org) or `mod_wsgi`. Please read the *Service management* (page 5) section.

Check the version:

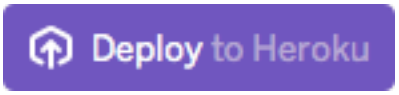
```
$ curl http://127.0.0.1:5000/about.json
{
  "api_v1_root": "/api/v1/",
  "version": "v0.1.9 - 05abfe1",
  "version_url": "https://github.com/monarc-project/stats-service/commits/05abfe1"
}
```

Install with a script:

```
$ curl -sSL https://raw.githubusercontent.com/monarc-project/stats-service/master/
↪contrib/install.sh | bash
```

### To Heroku

You can use this button:




(https://heroku.com/deploy?template=https://github.com/monarc-project/stats-service) or via command line:

```
$ git clone https://github.com/monarc-project/stats-service
$ cd stats-service/
$ heroku create --region eu <name-of-your-instance>
$ heroku addons:add heroku-postgresql:hobby-dev
$ heroku buildpacks:add --index 1 heroku/python
$ heroku buildpacks:add --index 2 https://github.com/heroku/heroku-buildpack-nodejs
$ heroku config:set HEROKU=1
$ heroku config:set INSTANCE_URL=https://<name-of-your-instance>.herokuapp.com
$ heroku config:set FLASK_APP='runserver.py'
$ heroku config:set FLASK_ENV='development'
$ git push heroku master
```

Create a new client:

```
heroku run flask client_create --name <name-of-the-client> --role admin
```

All commands (*Command Line Interface* (page 8)) are available. Just prefix with `heroku run`.

#### From the Python Package Index

If you use this method not all functionalities will be working, for the moment.

MONARC Stats service is available on PyPI (https://pypi.org/project/statsservice).

```
$ pipx install statsservice
$ monarc-stats-service
 * Serving Flask app "statsservice.bootstrap" (lazy loading)
 * Environment: production
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

If you want to use a custom configuration file:

```
$ curl https://raw.githubusercontent.com/monarc-project/stats-service/master/instance/
↪production.py.cfg -o production.py
$ export STATS_CONFIG=~/production.py
```

### 1.1.3 Service management

Several solutions are available:

- *Daemon* (page 5)
- *screen* (page 6)
- *mod_wsgi* (page 6)

#### Daemon

Whichever way you installed the service, you can choose to use systemd to start it. Simply create a file `/etc/systemd/system/statsservice.service` with the following contents:

```
[Unit]
Description=MONARC Stats service
After=network.target

[Service]
User=monarc
Environment=FLASK_APP=runserver.py
Environment=FLASK_ENV=production
Environment=STATS_CONFIG=production.py
WorkingDirectory=/home/monarc/stats-service
ExecStart=/home/monarc/.poetry/bin/poetry run flask run
Restart=always
```

(continues on next page)

(continued from previous page)

```
[Install]
WantedBy=multi-user.target
```

You may need to adjust it a bit (for example if you want to use Gunicorn). After adding this file to your system, you can start the new systemd service with these commands:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable statsservice.service
$ sudo systemctl start statsservice
$ systemctl status statsservice.service
```

### Accessing logs

```
$ journalctl -u statsservice
```

to follow the logs:

```
$ journalctl -u statsservice -f
```

### screen

(the geeky way)

```
$ screen -S statsservice
$ export STATS_CONFIG=production.py
$ poetry run python runserver.py
$ CTRL+a d
[detached from 183221.statsservice]
```

Connect to the session:

```
$ screen -ls
There is a screen on:
        183221.statsservice      (02/25/21 10:56:59)     (Detached)
1 Socket in /var/run/screen/S-cedric.
$ screen -xS 183221.statsservice
$
```

### mod_wsgi

Create a file `/etc/apache2/sites-available/statsservice.monarc.lu.conf` with a content similar to:

```
<VirtualHost *:80>
        ServerName stats.monarc.lu

        ServerAdmin webmaster@localhost
        DocumentRoot /home/monarc/stats-service
```

(continues on next page)

(continued from previous page)

```
        WSGIDaemonProcess statsservice user=www-data group=www-data threads=5 python-
↪home=/home/monarc/.local/share/virtualenvs/statsservice-_tH16p6s/ python-path=/home/
↪monarc/stats-service
        WSGIScriptAlias / /home/monarc/stats-service/webserver.wsgi

        <Directory /home/monarc/stats-service>
            WSGIApplicationGroup %{GLOBAL}
            WSGIProcessGroup statsservice
            WSGIPassAuthorization On

            Options Indexes FollowSymLinks
            Require all granted
        </Directory>

        SetEnv STATS_CONFIG production.py


        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn
        CustomLog /var/log/apache2/stats-service/access.log combined
        ErrorLog /var/log/apache2/stats-service/error.log

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

And a file:

```
$ cat stats-service/webserver.wsgi
#! /usr/bin/env python

python_home = '/home/monarc/.local/share/virtualenvs/statsservice-_tH16p6s'

activate_this = python_home + '/bin/activate_this.py'
with open(activate_this) as file_:
    exec(file_.read(), dict(__file__=activate_this))

from runserver import application
```

### 1.1.4 Integration with MONARC and collect of the stats

The technical guide of MONARC provides information about the integration of Stats Service with MONARC (https://www.monarc.lu/documentation/technical-guide/#stats-service). Especially related to the configuration of the cron job (which triggers a PHP command) on the MONARC Front Office. The cron job is responsible for collecting local statistics and sending those statistics to the Stats Service.

## 1.2 Updates

Updating the application consists of performing the following operations:

```
$ cd stats-service/
$ git pull origin master --tags
$ poetry install
$ poetry run flask db upgrade
$ sudo systemctl restart statsservice.service
```

To date, no database migrations are required.

## 1.3 Command Line Interface

In this section commands are presented per categories:



### 1.3.1 Database

#### Creation of the database

```
$ flask db_create --help
Usage: flask db_create [OPTIONS]

  Create the database from configuration parameters.

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

#### Initialization of the database

```
$ flask db_init --help
Usage: flask db_init [OPTIONS]

  Initialize the database.

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

#### Drop all the database

```
$ flask db_empty --help
Usage: flask db_empty [OPTIONS]

  Empty the database.

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

Will ask for confirmation and eventually drop all collections.

### 1.3.2 Clients

#### Creating a client

```
$ flask client_create --help
Usage: flask client_create [OPTIONS]

  Create a new client.

Options:
  --name TEXT   Name of the client.  [required]
  --uuid TEXT   UUID of the client.
  --token TEXT  Token of the client.
  --role TEXT   Role of the client (user or admin).  [default: user]
  --help        Show this message and exit.
```

Actually a client name must be specified and is unique.

Example:

```
$ flask client_create --name CASES
UUID: fb9b4d21-4082-445c-a6e6-5d42a1cd9202
Name: CASES
Role: 1
Token: fB5odBNDwzgia7SRm_Q-7tuiLtvIVHBZ2yOc9MopNMWzzoxdrF9K2cBo8rgS4eP_0Xsr0E0QCA_
↪jsQyjhXGaaQ
Created at: 2020-07-15 09:27:51.701245
```

#### List all clients

```
$ flask client_list
UUID: 7bb21fc8-9617-4da5-a60a-fbccda8cc583
Name: CIRCL
Role: 1
Token: IR1KPdoh94m8aoCV5cuRU5ROKWXS8m6lE5M96gklf1Ar6wbrogL_bFnDdpV_
↪AMrejApVsGfyNecp8THBXy108w
Created at: 2020-07-23 07:56:16.556226

UUID: aaaded6e-4039-448a-93bb-7cd7a696bc15
Name: SMILE
Role: 1
Token: wwqREga9eZUVH-cz2N40voD7BkirP5K0tlkANqK-
↪cKYjVXjy4YgdhtDGJAuw1oHntH79OSm3OzleVZEO3GRCeA
```

(continues on next page)

(continued from previous page)

```
Created at: 2020-07-23 07:57:47.658965

UUID: f490f727-9a1a-404b-bb91-ae36f643c6fe
Name: CASES
Role: 1
Token: RFXhRd4fDN7jQifaoBe3wF6TdGts6GSic2ec8qH0dft8Z2k-Q4ulZBoa_
↪50CrrUC6rLYSAEwJZsGySpuuuounw
Created at: 2020-07-17 12:48:05.192735
```

#### Delete a client

```
$ flask client_delete --help
Usage: flask client_delete [OPTIONS]

  Delete the client specified with its UUID and all the related local stats.

Options:
  --uuid TEXT  UUID of the client to delete.
  -y, --yes    Automatically reply yes to the confirmation message for the
               deletion of the client.

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

### 1.3.3 Stats

#### Pushing data

Pushing data to a central stats server.

```
$ flask stats_push --help
Usage: flask stats_push [OPTIONS]

  Pushes the clients stats to the global stats server.

Options:
  --date-from [%Y-%m-%d]  Only stats more recent than this date will be
                          pushed. Default value is 3 months before today.

  --date-to [%Y-%m-%d]    Only stats older than this date will be pushed.
                          Default value is today.

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

This command can be executed for example with cron.

The address of the central stats server must be specified in the configuration file.

The same remote token can be used to push the stats from different local clients.

#### Pulling data

Pulling data from a central stats server.

```
$ flask stats_pull --help
Usage: flask stats_pull [OPTIONS]

  Pull stats from an other stats instance for the local client specified in
  parameter.

Options:
  --client-uuid TEXT        Local client uuid
  --token TEXT       Client token on remote side  [required]
  --stats-type TEXT  Type of the stats to import (risk, vulnerability,
                     threat).  [required]

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

#### Delete stats

```
$ flask stats_delete --help
Usage: flask stats_delete [OPTIONS]

Delete the stats of a local client.

Options:
--client-uuid TEXT  UUID of the client related to the stats.
-y, --yes           Automatically reply yes to the confirmation message.
--help              Show this message and exit.
```

#### Purging stats

```
$ flask stats_purge --help
Usage: flask stats_purge [OPTIONS]

Delete the stats older than the number of months specified in parameter.

Options:
--nb-month INTEGER  Age (in months) of the stats to purge.
--help              Show this message and exit.
```

### 1.3.4 Interactions with MOSP

```
$ flask mosp_is_object_published --help
Usage: flask mosp_is_object_published [OPTIONS]

  Check if an object has been published on MOSP. Returns a boolean.

Options:
  --uuid TEXT    UUID of the object  [required]
  -v, --verbose  Display the object
  --help         Show this message and exit.
```

Examples:

```
$ flask mosp_is_object_published --uuid f3caa83b-28fb-49fd-b7ad-6e4cd1aaad06
False
$ flask mosp_is_object_published --uuid f3caa83b-28fb-49fd-b7ad-6e4cd1aaad07
True
$ flask mosp_is_object_published --uuid f3caa83b-28fb-49fd-b7ad-6e4cd1aaad07 -v
{
    "data": [
        {
            "description": "Mobile Mitigations  from MITRE ATT&CK® \r\n© 2020 The␣
↪MITRE Corporation. This work is reproduced and distributed with the permission of␣
↪The MITRE Corporation.",
            "json_object": {
                "authors": [
                    "MITRE ATT&CK®"
                ],
                "label": "MITRE ATT&CK - Mobile Mitigations",
                "language": "EN",
                "refs": [
                    "https://attack.mitre.org/mitigations/mobile/"
                ],
                "uuid": "f3caa83b-28fb-49fd-b7ad-6e4cd1aaad07",
                "values": [
                    {
                        "code": "M1013 - Application Developer Guidance",
                        "description": "This mitigation describes any guidance or␣
↪training given to developers of applications to avoid introducing security␣
↪weaknesses that an adversary may be able to take advantage of.",
                        "importance": 0,
                        "uuid": "90624dfc-21b6-4172-8848-a4042860656b"
                    },
                    {
                        "code": "M1005 - Application Vetting",
                        "description": "Enterprises can vet applications for␣
↪exploitable vulnerabilities or unwanted (privacy-invasive or malicious) behaviors.␣
↪Enterprises can inspect applications themselves or use a third-party service.",
                        "importance": 0,
                        "uuid": "7fd9df45-7351-420c-8116-57d48fa23c40"
                    },
                    {
                        "code": "M1002 - Attestation",
                        "description": "Enable remote attestation capabilities when␣
↪available (such as Android SafetyNet or Samsung Knox TIMA Attestation) and prohibit␣
↪devices that fail the attestation from accessing enterprise resources.",
                        "importance": 0,
                        "uuid": "5617161e-a40d-461a-ae8e-6a0650392e3a"
                    },
                    {
                        "code": "M1007 - Caution with Device Administrator Access",
                        "description": "Warn device users not to accept requests to␣
↪grant Device Administrator access to applications without good reason.",
                        "importance": 0,
                        "uuid": "63138250-3821-45f3-a820-55d0ffa30367"
                    },
                    {
                        "code": "M1010 - Deploy Compromised Device Detection Method",
                        "description": "A variety of methods exist that can be used␣
↪to enable enterprises to identify compromised (e.g. rooted/jailbroken) devices,␣
↪whether using security mechanisms built directly into the device, third-party␣
↪mobile security applications, enterprise mobility management (EMM)/mobile device␣
↪management (MDM) capabilities, or other methods. Some methods may be trivial to␣
↪evade while others may be more sophisticated.",
```

(continues on next page)

(continued from previous page)


```
                        "importance": 0,
                        "uuid": "6501d616-1a60-4b38-a40a-847ad5d28058"
                    },
                    {
                        "code": "M1009 - Encrypt Network Traffic",
                        "description": "Application developers should encrypt all of␣
↪their application network traffic using the Transport Layer Security (TLS) protocol␣
↪to ensure protection of sensitive data and deter network-based attacks. If desired,␣
↪application developers could perform message-based encryption of data before␣
↪passing it for TLS encryption.",
                        "importance": 0,
                        "uuid": "c591b8fd-5f57-4064-b5c5-f0acd38ae41f"
                    },
                    {
                        "code": "M1012 - Enterprise Policy",
                        "description": "An enterprise mobility management (EMM), also␣
↪known as mobile device management (MDM), system can be used to provision policies␣
↪to mobile devices to control aspects of their allowed behavior.",
                        "importance": 0,
                        "uuid": "b141135f-2c2f-4588-9d4c-6c7abd243e23"
                    },
                    {
                        "code": "M1014 - Interconnection Filtering",
                        "description": "In order to mitigate Signaling System 7 (SS7)␣
↪exploitation, the Communications, Security, Reliability, and Interoperability␣
↪Council (CSRIC) describes filtering interconnections between network operators to␣
↪block inappropriate requests.",
                        "importance": 0,
                        "uuid": "6066f816-7914-4228-96b6-155f4501d70c"
                    },
                    {
                        "code": "M1003 - Lock Bootloader",
                        "description": "On devices that provide the capability to␣
↪unlock the bootloader (hence allowing any operating system code to be flashed onto␣
↪the device), perform periodic checks to ensure that the bootloader is locked.",
                        "importance": 0,
                        "uuid": "148c35e1-7837-42a2-9884-4e475a48e6a3"
                    },
                    {
                        "code": "M1001 - Security Updates",
                        "description": "Install security updates in response to␣
↪discovered vulnerabilities.",
                        "importance": 0,
                        "uuid": "057adb3d-1eeb-4f04-a9c6-c08b514bc785"
                    },
                    {
                        "code": "M1004 - System Partition Integrity",
                        "description": "Ensure that Android devices being used␣
↪include and enable the Verified Boot capability, which cryptographically ensures␣
↪the integrity of the system partition.",
                        "importance": 0,
                        "uuid": "daa42611-836d-464e-aab5-80d41da314cf"
                    },
                    {
                        "code": "M1006 - Use Recent OS Version",
                        "description": "New mobile operating system versions bring␣
↪not only patches against discovered vulnerabilities but also often bring security␣
↪architecture improvements that provide resilience against potential vulnerabilities␣
↪or weaknesses that have not yet been discovered. They may also bring improvements␣
↪that block use of observed adversary techniques.",
```


(continues on next page)

(continued from previous page)


```
                        "importance": 0,
                        "uuid": "f4bbe273-dc6c-4b5d-8c66-286effded2c7"
                    },
                    {
                        "code": "M1011 - User Guidance",
                        "description": "Describes any guidance or training given to␣
↪users to set particular configuration settings or avoid specific potentially risky␣
↪behaviors.",
                        "importance": 0,
                        "uuid": "8f023e31-b83d-4323-ba0e-888ec025b35f"
                    }
                ],
                "version": 6.3
            },
            "last_updated": "2020-05-27T09:54:06.727943",
            "name": "MITRE ATT&CK - Mobile Mitigations "
        }
    ],
    "metadata": {
        "count": "1",
        "limit": "10",
        "offset": "0"
    }
}
True
```


## 1.4 Logging

Format of the log messages:

```
log_format="%(asctime)s %(levelname)s %(name)s %(funcName)s %(lineno)s: %(message)s"
```

Example:

```
2020-09-02 11:19:43,894 ERROR statsservice.api.v1.stats post 248: Duplicate stats:␣
↪fecea306-2b0e-4129-b34d-2a8876b1fede
```

You can define the path of the log file in the configuration file. The default is `./var/stats.log`. If not specified `sys.stderr` will be used.

### 1.4.1 Modules with logging

You can expect log messages from the following modules:

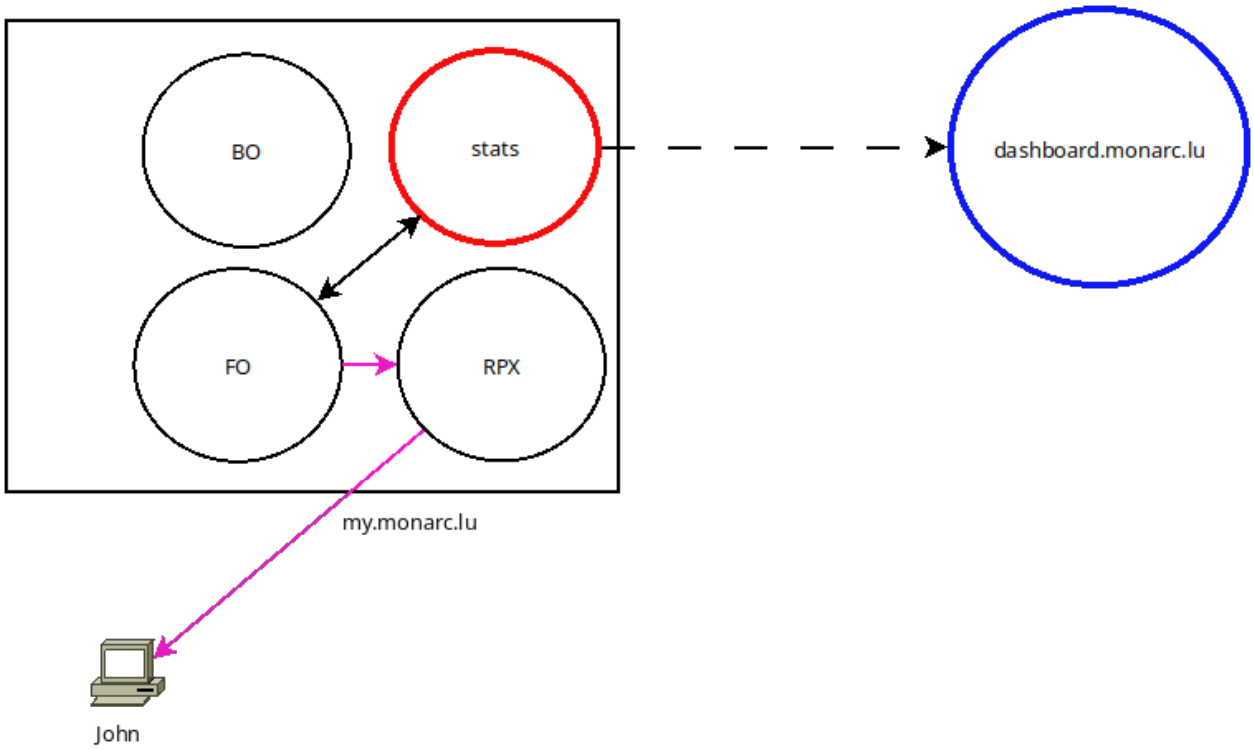
- statsservice.api.v1.client
- statsservice.api.v1.stats
- statsservice.api.v1.processed
- statsservice.commands.stats

# 2 Conceptual considerations

## 2.1 Architecture

These architecture diagrams presents the idea behind the decentralized nature of the Stats Service API.

### 2.1.1 Scenario 1




The Stats Service is installed on a dedicated server, gathering stats sent by the front office (FO). It can also return the aggregated stats to the MONARC backend (FO) for the dashboard of the MONARC users (with the CEO role).

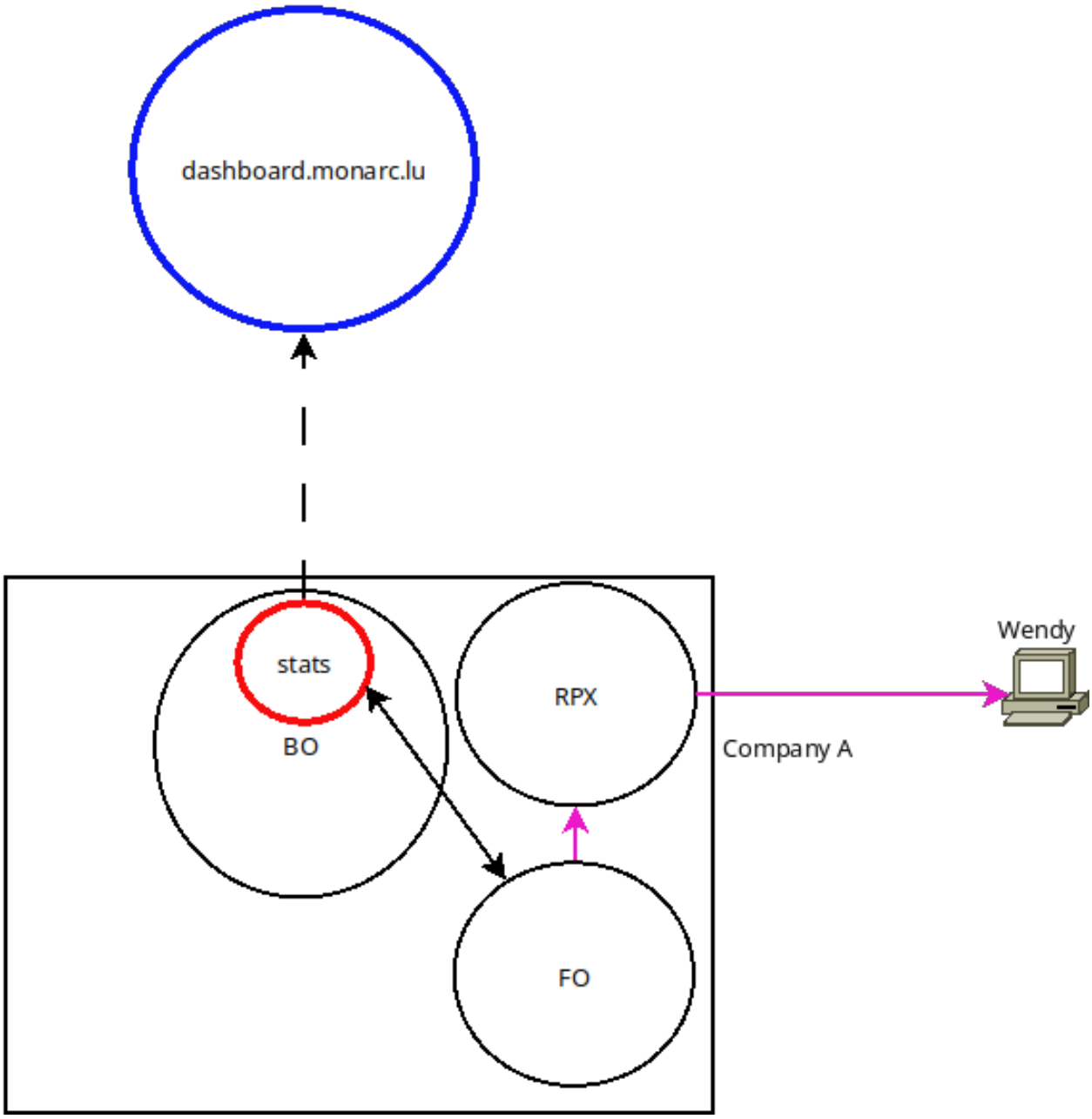
---

**Note:** The Stats Service only communicates with the backend of MONARC thanks to *its API* (page 19).

---

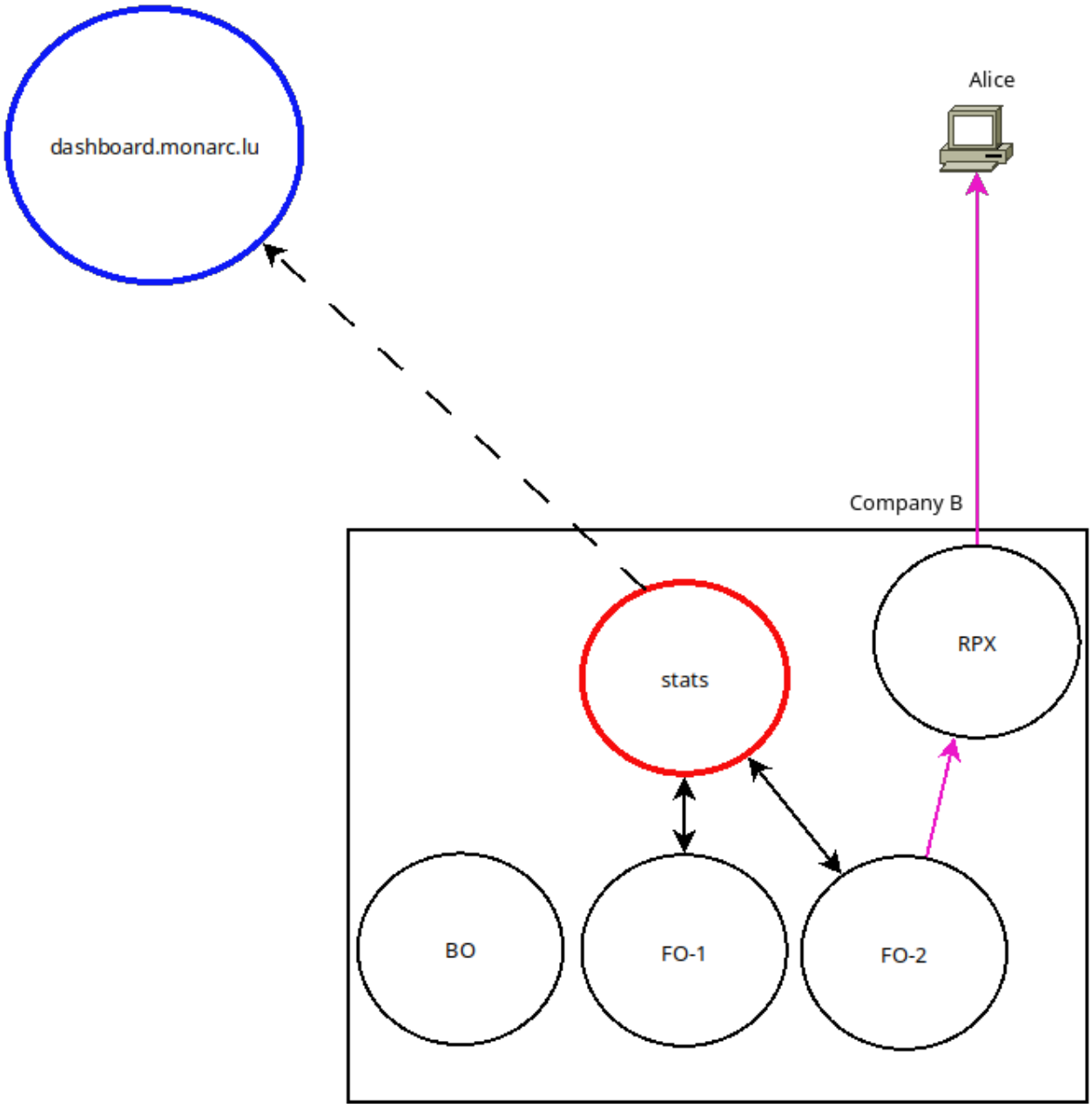
The stats collection (from the FO to the stats node) can be triggered with a cron job.

### 2.1.2 Scenario 2



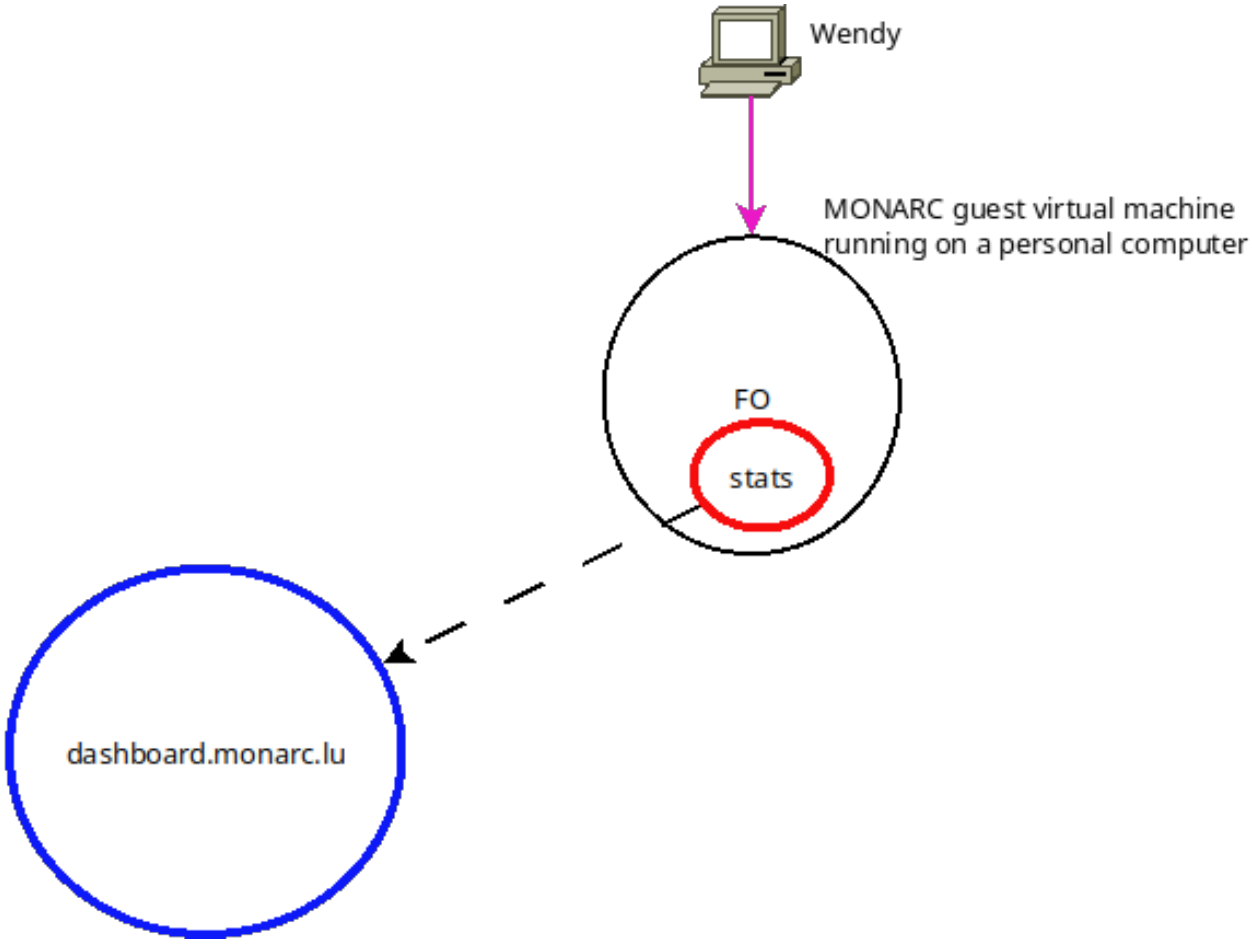

In the company A, the difference is that Stats Service API (*stats*) is installed on the same server where the MONARC back office is deployed. Not on on a dedicated server. The server hosting the back office of MONARC is a good choice.

### 2.1.3 Scenario 3




*stats* collects data from the two MONARC front offices of the company B. These aggregated data can be used for the dashboard for the CEO role of MONARC client instances in this company. But also, in the future, for the MONARC back office.

### 2.1.4 Scenario 4




Wendy is simply connected to a MONARC front office deployed in a local Virtual Box instance. Stats Service (*stats*) is running next to MONARC in this virtual machine.

### 2.1.5 Important notes

---

**Note:** In all scenario, **locally collected** statistics (in *stats* node) can optionally be sent to dashboard.monarc.lu (https://dashboard.monarc.lu) which is a global instance. It's the same software. You can configure the global dashboard statistics **sharing** option in the General Settings / Sharing statistics (https://www.monarc.lu/documentation/user-guide/#global-dashboard) view of your MONARC instance (see here (https://www.monarc.lu/documentation/user-guide/images/GlobalDashboardGlobalSetting.png)).

However, stats must always be collected in your local Stats Service instance. This is required for the proper functioning of the global dashboard of your MONARC instance.

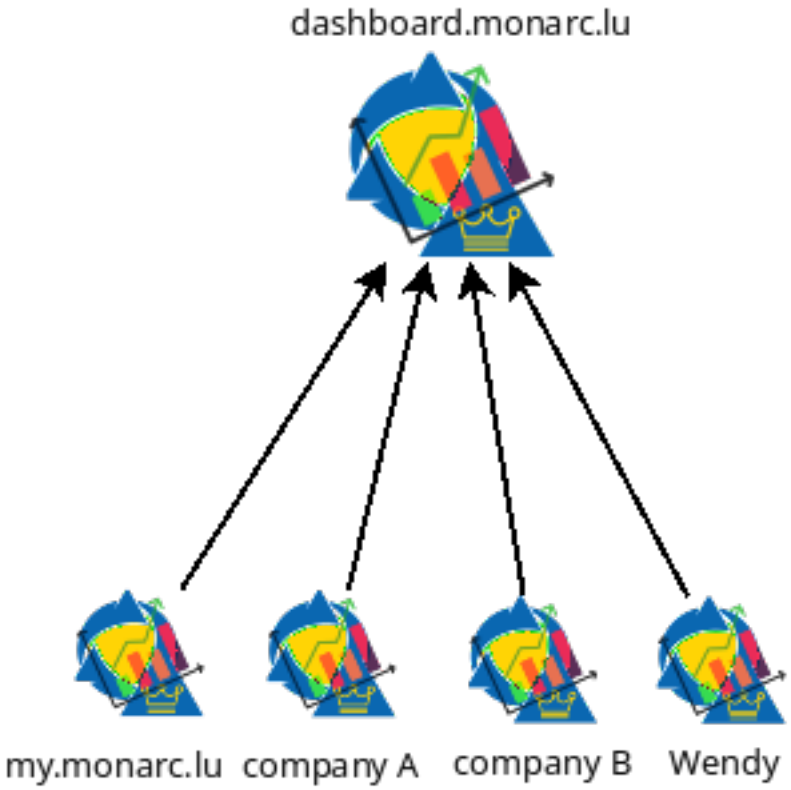
---

---

**Note:** It is as well possible to run your own alternative to dashboard.monarc.lu. And then you can configure the variable *REMOTE_STATS_SERVER* of your local Stats Service instance with the address of your "global" instance.

---

### 2.1.6 Organization level




A company has the possibility to push stats from selected clients with the same account on, for example, dashboard.monarc.lu. The service behind dashboard.monarc.lu is operated by CASES Luxembourg and aims to provide relevant data about the current cybersecurity trends, focused in the Luxembourg area.

### 2.1.7 Integration with external services

#### MOSP

A Stats Service instance is able to query MOSP.

# 3 Blueprints

## 3.1 API v1

Blueprint dedicated to the main API. It is composed of two namespaces, *client* and *stats*, that are used for the endpoints `api/v1/stats` and `api/v1/client`.

New namespaces are easily plugable via the file: `api/v1/__init__.py`.

### 3.1.1 Security model

First, an overview of the security model.

Clients needs to have an account on the Stats Service and a token in order to submit new stats and to query the API. New clients can be created with the dedicated command.

The token must be submitted in the headers of the requests with the key `X-API-KEY`, as explained below. It is not needed to submit an additional id, tokens are unique.

### 3.1.2 OpenAPI Specification

Specification also available at https://dashboard.monarc.lu/api/v1 (https://dashboard.monarc.lu/api/v1/).


```
{
    "swagger": "2.0",
    "basePath": "/api/v1",
    "paths": {
        "/client/": {
            "post": {
                "responses": {
                    "201": {
                        "description": "Success",
                        "schema": {
                            "$ref": "#/definitions/Clients"
                        }
                    }
                },
                "summary": "Create a new client",
                "operationId": "client_create",
                "parameters": [
                    {
                        "name": "payload",
                        "required": true,
                        "in": "body",
                        "schema": {
                            "$ref": "#/definitions/Clients"
                        }
                    },
                    {
                        "name": "X-Fields",
                        "in": "header",
                        "type": "string",
                        "format": "mask",
                        "description": "An optional fields mask"
                    }
                ],
                "tags": [
                    "client"
                ]
            }
        },
        "/client/me": {
            "get": {
                "responses": {
                    "200": {
                        "description": "Success",
                        "schema": {
                            "$ref": "#/definitions/Clients"
                        }
                    }
```


(continues on next page)

(continued from previous page)

```
                },
                "operationId": "client_get",
                "parameters": [
                    {
                        "name": "X-Fields",
                        "in": "header",
                        "type": "string",
                        "format": "mask",
                        "description": "An optional fields mask"
                    }
                ],
                "tags": [
                    "client"
                ]
            },
            "patch": {
                "responses": {
                    "201": {
                        "description": "Success",
                        "schema": {
                            "$ref": "#/definitions/Clients"
                        }
                    }
                },
                "operationId": "client_patch",
                "parameters": [
                    {
                        "name": "payload",
                        "required": true,
                        "in": "body",
                        "schema": {
                            "$ref": "#/definitions/Clients"
                        }
                    },
                    {
                        "name": "X-Fields",
                        "in": "header",
                        "type": "string",
                        "format": "mask",
                        "description": "An optional fields mask"
                    }
                ],
                "tags": [
                    "client"
                ]
            }
        },
        "/stats/": {
            "get": {
                "responses": {
                    "401": {
                        "description": "Authorization needed"
                    },
                    "200": {
                        "description": "Success",
                        "schema": {
                            "type": "array",
```

(continues on next page)

(continued from previous page)

```
                    "items": {
                        "$ref": "#/definitions/StatsList"
                    }
                }
            }
        },
        "summary": "List all stats",
        "operationId": "list_stats",
        "parameters": [
            {
                "name": "payload",
                "required": true,
                "in": "body",
                "schema": {
                    "type": "object",
                    "properties": {
                        "anr": {
                            "type": "string"
                        },
                        "type": {
                            "type": "string"
                        },
                        "group_by_anr": {
                            "type": "integer"
                        },
                        "date_from": {
                            "type": "string"
                        },
                        "date_to": {
                            "type": "string"
                        },
                        "anrs": {
                            "type": "string"
                        },
                        "get_last": {
                            "type": "boolean"
                        },
                        "offset": {
                            "type": "integer"
                        },
                        "limit": {
                            "type": "integer"
                        }
                    }
                }
            },
            {
                "name": "X-Fields",
                "in": "header",
                "type": "string",
                "format": "mask",
                "description": "An optional fields mask"
            }
        ],
        "tags": [
            "stats"
        ]
```

(continues on next page)

(continued from previous page)

```
        },
        "post": {
            "responses": {
                "401": {
                    "description": "Authorization needed"
                },
                "201": {
                    "description": "Success",
                    "schema": {
                        "type": "array",
                        "items": {
                            "$ref": "#/definitions/StatsList"
                        }
                    }
                }
            },
            "summary": "Create a new stats",
            "operationId": "create_stats",
            "parameters": [
                {
                    "name": "payload",
                    "required": true,
                    "in": "body",
                    "schema": {
                        "type": "array",
                        "items": {
                            "$ref": "#/definitions/Stats"
                        }
                    }
                },
                {
                    "name": "X-Fields",
                    "in": "header",
                    "type": "string",
                    "format": "mask",
                    "description": "An optional fields mask"
                }
            ],
            "tags": [
                "stats"
            ]
        }
    },
    "/stats/processed/": {
        "get": {
            "responses": {
                "200": {
                    "description": "Success",
                    "schema": {
                        "type": "array",
                        "items": {
                            "$ref": "#/definitions/Result"
                        }
                    }
                }
            },
            "summary": "Return the result of the processor",
```

(continues on next page)

(continued from previous page)

```
                "operationId": "processing_list",
                "parameters": [
                    {
                        "name": "payload",
                        "required": true,
                        "in": "body",
                        "schema": {
                            "type": "object",
                            "properties": {
                                "type": {
                                    "type": "string"
                                },
                                "processor": {
                                    "type": "string"
                                },
                                "processor_params": {
                                    "type": "string"
                                },
                                "anrs": {
                                    "type": "string"
                                },
                                "date_from": {
                                    "type": "string"
                                },
                                "date_to": {
                                    "type": "string"
                                },
                                "local_stats_only": {
                                    "type": "integer"
                                }
                            }
                        }
                    },
                    {
                        "name": "X-Fields",
                        "in": "header",
                        "type": "string",
                        "format": "mask",
                        "description": "An optional fields mask"
                    }
                ],
                "tags": [
                    "processed"
                ]
            }
        },
        "/stats/processed/list": {
            "get": {
                "responses": {
                    "200": {
                        "description": "Success"
                    }
                },
                "summary": "Return the list of available processors with their␣
↪description",
                "operationId": "get_processor_list",
                "tags": [
```

(continues on next page)

(continued from previous page)

```
                    "processed"
                ]
            }
        },
        "/stats/{anr}": {
            "parameters": [
                {
                    "in": "query",
                    "description": "The stats identifier",
                    "name": "uuid",
                    "type": "string"
                },
                {
                    "name": "anr",
                    "in": "path",
                    "required": true,
                    "type": "string"
                }
            ],
            "delete": {
                "responses": {
                    "404": {
                        "description": "Stats not found"
                    },
                    "204": {
                        "description": "Stats deleted"
                    }
                },
                "summary": "Delete stats by provided anr",
                "operationId": "delete_stats",
                "tags": [
                    "stats"
                ]
            },
            "get": {
                "responses": {
                    "404": {
                        "description": "Stats not found"
                    },
                    "200": {
                        "description": "Success",
                        "schema": {
                            "$ref": "#/definitions/Stats"
                        }
                    }
                },
                "summary": "Fetch a given resource by anr",
                "operationId": "get_stats",
                "parameters": [
                    {
                        "name": "X-Fields",
                        "in": "header",
                        "type": "string",
                        "format": "mask",
                        "description": "An optional fields mask"
                    }
                ],
```

(continues on next page)

(continued from previous page)


```
                "tags": [
                    "stats"
                ]
            }
        }
    },
    "info": {
        "title": "MONARC Stats service - API v1",
        "version": "1.0",
        "description": "API v1 of the MONARC Stats service.",
        "license": {
            "name": "GNU Affero General Public License version 3",
            "url": "https://www.gnu.org/licenses/agpl-3.0.html"
        }
    },
    "produces": [
        "application/json"
    ],
    "consumes": [
        "application/json"
    ],
    "securityDefinitions": {
        "apikey": {
            "type": "apiKey",
            "in": "header",
            "name": "X-API-KEY"
        }
    },
    "security": [
        {
            "apikey": []
        }
    ],
    "tags": [
        {
            "name": "client",
            "description": "client related operations"
        },
        {
            "name": "stats",
            "description": "stats related operations"
        },
        {
            "name": "processed",
            "description": "Processing related operations on stats data."
        }
    ],
    "definitions": {
        "Clients": {
            "properties": {
                "name": {
                    "type": "string",
                    "description": "The client name."
                },
                "token": {
                    "type": "string",
                    "description": "The token of the client.",
```


(continues on next page)

(continued from previous page)

```
                    "readOnly": true
                },
                "role": {
                    "type": "string",
                    "description": "The client role.",
                    "readOnly": true
                },
                "is_sharing_enabled": {
                    "type": "boolean",
                    "description": "If the statistics sharing is enabled or not.",
                    "readOnly": true
                }
            },
            "type": "object"
        },
        "Stats": {
            "properties": {
                "uuid": {
                    "type": "string",
                    "description": "The stats unique identifier",
                    "readOnly": true
                },
                "anr": {
                    "type": "string",
                    "description": "The ANR UUID related to this stats."
                },
                "type": {
                    "type": "string",
                    "description": "The type of this stats (risk, vulnerability,␣
↪threat, cartography or compliance)."
                },
                "date": {
                    "type": "string",
                    "format": "date",
                    "description": "The stats date in format 'Y-m-d'"
                },
                "data": {
                    "type": "object",
                    "description": "The stats as a dynamic JSON object."
                }
            },
            "type": "object"
        },
        "StatsList": {
            "properties": {
                "metadata": {
                    "description": "Metada related to the result.",
                    "allOf": [
                        {
                            "$ref": "#/definitions/metadata"
                        }
                    ]
                },
                "data": {
                    "type": "array",
                    "description": "List of stats objects.",
                    "items": {
```

(continues on next page)

(continued from previous page)


```
                        "$ref": "#/definitions/Stats"
                    }
                }
            },
            "type": "object"
        },
        "metadata": {
            "properties": {
                "count": {
                    "type": "string",
                    "description": "Total number of the items of the data.",
                    "readOnly": true
                },
                "offset": {
                    "type": "string",
                    "description": "Position of the first element of the data from␣
↪the total data amount.",
                    "readOnly": true
                },
                "limit": {
                    "type": "string",
                    "description": "Requested limit data.",
                    "readOnly": true
                }
            },
            "type": "object"
        },
        "Result": {
            "properties": {
                "type": {
                    "type": "string",
                    "description": "Type of the processed stats data (risk,␣
↪vulnerability, threat, cartography or compliance)."
                },
                "processor": {
                    "type": "string",
                    "description": "Processor used for the stats data processing."
                },
                "data": {
                    "type": "object",
                    "description": "Result of the selected processor applied to the␣
↪resulting stats."
                }
            },
            "type": "object"
        }
    },
    "responses": {
        "ParseError": {
            "description": "When a mask can't be parsed"
        },
        "MaskError": {
            "description": "When any error occurs on mask"
        }
    }
}
```

### 3.1.3 Detailed examples with the API

#### Getting stats for a client

```
$ curl -X GET "http://127.0.0.1:5000/api/v1/stats" -H  "accept: application/json" -H
↪"X-API-KEY: rddPRk9_t-Z4GOgmY2UL2blKB1DxWB_0yhDlqcsF9p63eXs-
↪oLCdm2c9YgP7cOqGz71GK1tc8lrCenD8AvEr-g"
{
  "metadata": {
    "count": "0",
    "offset": "0",
    "limit": "0"
  },
  "data": []
}
```

Specify a type of stats:

```
$ curl --location --request GET \
-H 'X-API-KEY: rddPRk9_t-Z4GOgmY2UL2blKB1DxWB_0yhDlqcsF9p63eXs-
↪oLCdm2c9YgP7cOqGz71GK1tc8lrCenD8AvEr-g' \
-H "Content-type: application/json" \
-H "Accept: application/json" \
-d '{"type":"threat"}' \
'http://127.0.0.1:5000/api/v1/stats'
```

#### Invoking a processor

Processors are utilities to process stats data. Processors are currently defined in `statsservice/lib/processors.py`. They operate on different kind of stats:

- risks;
- threats;
- vulnerabilities.

They can do a lot of things, like evaluate different averages based on user requirements. The result of the processor is sent in the response from the API.

You can get the list of available processors:

```
$ curl http://127.0.0.1:5000/api/v1/stats/processed/list
[
    {
        "name": "risk_averages",
        "description": "Evaluates the averages for the risks. Averages are evaluated␣
↪per categories\n    (current/residual, informational/operational, low/medium/high)."
    },
    {
        "name": "risk_averages_on_date",
        "description": "Evaluates the averages for the risks per date. Averages are␣
↪evaluated per categories\n    (current/residual, informational/operational, low/
↪medium/high).\n    Supported parameters:\n    - risks_type: informational or␣
↪operational\n    - risks_state: current or residual."
    },
    {
```

(continues on next page)

(continued from previous page)

```
        "name": "threat_average_on_date",
        "description": "Aggregation and average of threats per date for each threat␣
↪(accross all risk\n    analysis).\n    "
    },
    {
        "name": "vulnerability_average_on_date",
        "description": "Aggregation and average of vulnerabilities per date for each␣
↪vulnerability\n    (accross all risk analysis).\n    "
    }
]
```

Some processors can use dedicated parameters. For this purpose the parameter `processor_params` can be used to transfer these parameters directly to the concerned processor. These parameters will affect the behaviour of the processor.

For example you might want to call the processor `risk_averages_on_date` but you want that this processor only evaluates the averages for *residual risks* that are also *operational risks* (and not informational risks). The request will look like:

```
curl -X GET "http://127.0.0.1:5000/api/v1/stats/processed/" -H  "accept: application/
↪json" -H  "Content-Type: application/json" -d "{  \"type\": \"risk\",  \"processor\
↪": \"risk_averages_on_date\",  \"local_stats_only\": 0, \"date_from\":\"2020-06-12\
↪",\"processor_params\": {\"risks_type\":\"operational\",\"risks_state\":\"residual\
↪"}}"
```

Internally the processor `processor_params` will honor the value provided with the parameters `risks_type` and `risks_state`.

- `risks_type` can be *informational* or *operational*;
- `risks_state` can be *current* or *residual*.

Generally, you can get information about a processor:

```
$ echo -e `curl -s http://127.0.0.1:5000/api/v1/stats/processed/list | jq '.[] |␣
↪select(.name=="risk_averages_on_date") | .description'`
"Evaluates the averages for the risks per date. Averages are evaluated per categories
 (current/residual, informational/operational, low/medium/high).
 Supported parameters:
 - risks_type: informational or operational
 - risks_state: current or residual."
```

## 3.2 Blueprint stats

The goal of this blueprint is to return formatted custom stats. It is in read-only mode with only public routes (no authentication required).

### 3.2.1 Endpoints

- /stats. At that time this route simply returns a HTML file. Some charts can be displayed for example with the data from the following routes.
- /stats/threats.json
- /stats/vulnerabilities.json

- /stats/risks.json

### Threats

```
$ curl http://127.0.0.1:5000/stats/threats.json?processor=threat_average_on_date&
↪days=100
{
  "b402d4e0-4576-11e9-9173-0800277f0571": {
    "2020-07-13": {
      "averageRate": 2.32,
      "count": 10.0,
      "maxRisk": 35.5
    },
    "2020-07-14": {
      "averageRate": 3.0,
      "count": 8.0,
      "maxRisk": 34.0
    },
    "2020-07-27": {
      "averageRate": 3.0,
      "count": 8.0,
      "maxRisk": 34.0
    },
    "2020-07-28": {
      "averageRate": 2.82,
      "count": 12.0,
      "maxRisk": 36.0
    },
    "2020-07-29": {
      "averageRate": 3.0,
      "count": 8.0,
      "maxRisk": 34.0
    },
    "2020-07-31": {
      "averageRate": 2.82,
      "count": 6.0,
      "maxRisk": 36.0
    }
  },
  "b402d523-4576-11e9-9173-0800277f0571": {
    "2020-07-13": {
      "averageRate": 2.87,
      "count": 8.0,
      "maxRisk": 45.0
    },
    "2020-07-14": {
      "averageRate": 2.87,
      "count": 8.0,
      "maxRisk": 45.0
    },
    ...
  }
  ...
}
```

- if the parameter `days` is not specified the default value is 365.

- `threat_average_on_date` is also the default processor.

### Vulnerabilities

```
$ curl http://127.0.0.1:5000/stats/vulnerabilities.json?processor=vulnerability_
↪average_on_date&days=100
{
  "69fbfe14-4591-11e9-9173-0800277f0571": {
    "2020-08-01": {
      "averageRate": 2.0,
      "count": 3.0,
      "maxRisk": 18.0
    },
    "2020-08-03": {
      "averageRate": 2.0,
      "count": 3.0,
      "maxRisk": 18.0
    }
  },
  "69fbfe5f-4591-11e9-9173-0800277f0571": {
    "2020-08-01": {
      "averageRate": 1.0,
      "count": 1.0,
      "maxRisk": 6.0
      },
      ...
    }
    ...
  }
```

### Risks

```
$ curl http://127.0.0.1:5000/stats/risks.json?processor=risk_averages_on_date
{
  "current": {
    "informational": {
      "High risks": {
        "2020-10-01": 8.0,
        "2020-10-12": 8.0
      },
      "Low risks": {
        "2020-10-01": 66.0,
        "2020-10-12": 12.0
      },
      "Medium risks": {
        "2020-10-01": 27.0,
        "2020-10-12": 27.0
      }
    },
    "operational": {
      "High risks": {
        "2020-10-01": 10.0,
        "2020-10-12": 10.0
      },
```

(continues on next page)

(continued from previous page)


```
      "Low risks": {
        "2020-10-01": 18.0,
        "2020-10-12": 18.0
      },
      "Medium risks": {
        "2020-10-01": 0.0,
        "2020-10-12": 0.0
      }
    }
  },
  "residual": {
    "informational": {
      "High risks": {
        "2020-10-01": 1.0,
        "2020-10-12": 1.0
      },
      "Low risks": {
        "2020-10-01": 74.0,
        "2020-10-12": 74.0
      },
      "Medium risks": {
        "2020-10-01": 26.0,
        "2020-10-12": 26.0
      }
    },
    "operational": {
      "High risks": {
        "2020-10-01": 0.0,
        "2020-10-12": 0.0
      },
      "Low risks": {
        "2020-10-01": 28.0,
        "2020-10-12": 28.0
      },
      "Medium risks": {
        "2020-10-01": 0.0,
        "2020-10-12": 0.0
      }
    }
  }
}
```


```
$ curl http://127.0.0.1:5000/stats/risks.json?processor=risk_averages
```


```
{
  "current": {
    "informational": {
      "High risks": 8.0,
      "Low risks": 30.0,
      "Medium risks": 27.0
    },
    "operational": {
      "High risks": 10.0,
      "Low risks": 18.0,
      "Medium risks": 0.0
    }
  },
```


(continues on next page)

(continued from previous page)


```
  "residual": {
    "informational": {
      "High risks": 1.0,
      "Low risks": 74.0,
      "Medium risks": 26.0
    },
    "operational": {
      "High risks": 0.0,
      "Low risks": 28.0,
      "Medium risks": 0.0
    }
  }
}
```