boi-Das on-premise v1.12.5

Deployment and Working Instructions

Release	Date	Description	Author	Reviewer	Approver
2.15	05/08/2021	Release	IPA	ACR	MSY

1. Introduction

boi-Das is a Case-Management tool developed and offered by Veridas for the review of the validation processes by back-office teams.

It consists of a GUI-based web service which shows all the validation processes carried on by using the vali-Das' API and an API which allows to retrieve the validations data as well as modify certain parts of them.

Both the UI and the API allow to take decisions and actions regarding the veracity of the ID document and biometrics data and results obtained on the validation processes, such as modify their status to accepted or rejected, and edit the OCR personal data.

This product is released as a configurable docker solution comprising three parts, the boi-Das application, a PSQL database and an NGINX web server which, among others, serves the static files.

2. Changelog

boi-Das v1.12.5

General

- **Added**
  - Expire session automatically after a certain period of time. The default value is 12 hours, but it is possible to modify using the new environment variable SESSION_TIMEOUT
- **Changed**
  - Upgrade boi-Das docker services to use NGINX version 1.19 and Ubuntu 20.04
  - Improve logging registry
  - Security improvements regarding HTTP protocol, TLS protocol, cookies management, docker services version and more.
  - Remove dependency with AVX instruction set

User Interface

- **Added**
  - Personal data (OCR text) edition: The user is able to modify the personal data on a pending validation and save the changes so that the modified data is available each time the validation is retrieved
  - New timestamp information is shown in the "Validation Information" tab
  - New searching engine with "null" string matching for empty fields

- **Changed**

- Rename “Contextual data” tab to “Validation Information”

- **Fixed**
  - Remove the limitation of exporting only ten validations at a time

API

- **Added**
  - New API endpoints which allow supervisor to manage contextual filters
  - New API endpoint which allows to obtain timestamp zip file

Export validations

- **Added**
  - Export now includes timestamp information
  - New zip file “timestamp.zip” included in the validation export package collecting all timestamp artifacts including hash

3. Docker Deployment Instructions

As commented before, boi-Das interfaces are configurable docker solutions which can be deployed easily by using docker commands.

First of all, to load the docker image from the provided tar file, the “docker load” command has to be run:

$ docker load -i boiDas-DockerImage-X.Y.Z.tar

Part of the boi-Das service configuration is hardcoded while other can be configured by means of environmental variables and logical data volumes. boi-Das service parts run on a Docker container which is built over:

- Ubuntu 20.04 LTS.
- Nginx proxy (with or without SSL).
- PostgreSQL database.

The first time this container runs, it assumes the database access given in the environmental variables is valid, uses these credentials to build and prepare the database, and also populates the database with an initial dump. For this purpose, this image imports a few data volumes from the running localhost machine, using them as persistent storage for configuration files. Others, as output logs, should be kept isolated. By doing it this way, the next time this container is created, it will use the configuration of the previous one, unless the user erased all files in the data volumes.

In order to achieve a proper deployment, it is important to be in a clear environment the first time this container is created. This way, we ensure the database and other files and directories are synchronized.

3.1. boi-Das Host (server) Requirements

3.1.1. Regular Usage

For use-cases with sustainable workload of up to 10x agents or supervisors and up to 10 validations per minute:

- Server with i5 2x core processor with minimum of 2.5 GHz and 8GB RAM

3.1.2. Intensive Usage

For workloads between 10 and 50x agents or supervisors and between 10 and 50 validation per minute:

- Server with i7 6x core processor with minimum of 2.5 GHz and 20GB RAM

For higher usage requirements, please ask your solution provider.

3.1.3. Common requirements

- Ubuntu (18.04 LTS or higher). This is the OS which Veridas supports and validates with each release, but as the product is docker-based, it should work in any Linux environment, specially Debian based distros like Red Hat Enterprise Linux (7.9 or higher)
- Docker and Logrotate applications installed
- A domain name (optional, but recommended)
- An SSL certificate associated to the domain which is used on the docker container deployment (optional, but recommended)
- PSQL database (it may be deployed as a docker container or use an existing database)
- Appropriate disk space depending on the system usage, considering a disk usage of around 30 MB per validation. It is also convenient to take into account that when validations are exported from the boi-Das dashboard, the zip files containing the validations are stored in the corresponding mounted volume, so if they are not deleted periodically, it also requires additional disk space.
- Ad-hoc boi-Das data backup service is out of the scope of this document, but its implementation highly recommended

3.1.4. Network configuration requirements

Boidas service requires connectivity to VeriSaaS APIs for retrieving the validations processed once confirmed. For this task, all the boidas instances require connectivity to TCP/443 of the following URLs:

EU-Sandbox: https://api-work.eu.veri-das.com/*
EU-Production: https://api.eu.veri-das.com/*
US-Production: https://api.us.veri-das.com/*

Please ensure that proper firewall/WAF rules are put in place to allow outgoing connections to the URL (not the host) depending on the environment to which it is connected.

In addition to this, if you are using videocall feature, another URL needs to be reachable from boidas instances:

EU-Sandbox: https://videocallmanager-work.eu.veri-das.com/*
EU-Production: https://videocallmanager.eu.veri-das.com/*
US-Production: https://videocallmanager.us.veri-das.com/*

The asterisk at the end of the URLs indicates that any path after that URL should be allowed.

If you have IP restriction enabled at your VeriSaaS account, please ensure your boidas instances uses a fixed IP address or IP subnet when reaching the Internet and it is included in the allowList of validas at cloud. If you have any questions, please reach out to Veridas Customer Support for clarification.

Finally, incoming connections to boidas URL and TCP/port (whichever was setup) should also be allowed in the Firewall/WAF rules to allow boidas users to reach its UI/API.

3.1.4.2 Agents and supervisors

- Using boi-Das with or without the video call feature, the host/ devices which agents and supervisors use to do the boidas validations review process, require to reach this hostnames and ports:
    - ${BOIDAS_HOSTNAME}:TCP/<NGINX_PORT>
- Using boi-Das with the video call feature, both the host where boidas is deployed and the host which the agents and supervisors are using to connect to boidas, require to have access to these hostnames and ports:
    - Any TCP/443, UDP/10000, TCP/4443

3.1.4.3 CRM

- IP CRM to boidas hostname:<NGINX_PORT>

3.1.4.4 End-Users

- IP end-user to any TCP/443, UDP/10000, TCP/4443

3.2. Database Host (server) Requirements for a Dedicated Machine Case

3.2.1. Non Intensive Usage

- Computer with i5 2 core processor with minimum frequency of 2.5 GHz and 4GB RAM

3.2.2. Intensive Usage

- Computer with i5 4 core processor with minimum frequency of 2.5 GHz and 6GB RAM

3.2.3. In Both Scenarios

- Appropriate disk space depending on the usage estimations and considering a growth of around 30 MB per validation process
- Minimum Postgres version supported is 9.6, being the 12.6 the recommended one. Postgres version 9.6 will reach the end of life in November 2021. Veridas recommends upgrading to the recommended version or to a one higher than 9.6 before that date, because support will not be available for this version after that date.
- There is no condition for the type of disks. SATA, SSD, SAS and other are valid, obtaining small performance difference depending of its features but with little impact on the global system performance
- RAID 1 at least is recommended

3.3. boi-Das end users Host (client) Requirements

3.3.1. Regular Usage with video call feature

- Desktop with i5 2x core processor with minimum of 2.5 GHz and 4GB RAM

4. Docker Container Configuration

4.1 boi-Das docker

4.1.1 General Configuration

The container creation procedure is configured by using many environmental variables which can be given to the docker *run* command or in a docker-compose.yml file. The default value of these variables is given after equals sign.

- VALIDAS_URL: ie. https://api.eu.veri-das.com/validas/v1 => This URL has to be set to the vali-Das URL which is deployed on the Veridas SaaS. If this is not correctly set, boi-Das will not be able to retrieve the validations. [Note: The URL will be different depending on it the environment is Production or Sandbox, and will be provided by Veridas]
- API_KEY: 'API_KEY' => This has to be set to the API KEY provided by Veridas when vali-Das documentation was given.
- DB_HOST: pgsql-boidas => Here the host where DB is deployed. Can be the container name or the IP of the host where the DB is deployed.
- TZ=Europe/Madrid => Desired Timezone.
- DB_NAME: boidas => Name of the database for boi-Das.
- DB_USER: boidas => Name of the user granted to DB_NAME with DB_PASS.
- DB_PASS: boidas => A dummy default password.
- ENABLE_SSL: TRUE => Indicates if the system requires SSL or not.
- PORT: 5080 => the port that the service container is going to expose. If this variable is not set, the exposed PORT is 8850 by default.
- BASE_URL => The URL conformed with the protocol + hostname + port (https://HOSTNAME:PORT), i.e. https://myawesomehostname:5081) where the API is deployed. This allows the service to know where it is being deployed, which is used for paths creation in API responses.
- MIGRATIONS: "yes" or "no" => the container needs to do database migrations to prepare the database with the last models and configurations. It is required to do the first time or when upgrading a new version.
- EMAIL_HOST_NAME: "smtp.mycompany.com" => SMTP server configured to send emails when agents forgot password to access
- EMAIL_HOST_PORT: "587" => SMTP port
- EMAIL_HOST_USER: "myloginaccount@mycompany.com" => SMTP login account
- EMAIL_HOST_PASSWORD: "xxx" => SMTP password
- DEFAULT_FROM_EMAIL: "no-reply@mycompany.com" => email account configured as FROM for sending emails
- ENABLE_ACTIVITY_REGISTRY: "yes" or "no" => If this variable is set to "yes", the validation related events involving user actions are saved and also shown in the activity tab on the boidas UI. Default value is set to "yes".
- MANDATORY_SEPBLAC_QUESTIONS: "yes" or "no" => If this variable is set to "yes", a set of questions related to both the document and the biometry verifications are shown in boidas validation detail screens, and must be answered to allow a validation to be approved or pre-approved. Answering these questions, allows to comply with the Sepblac regulation regarding the digital onboarding processes bor bank account oppening. Default value is set to "no".

- SECONDS_TIMEOUT_SPA: “600” => This is the period of time in seconds used to automatically log out a user if there is no activity. So, if this amount of seconds passes without any user activity on the interface, the boi-Das service automatically logs out the user. The default value is set to “600”.
- SESSION_TIMEOUT: “43200” => This is the period of time in seconds used to automatically expire the session. So, if this amount of seconds passes, the boi-Das service automatically expires the user session and the log in will be required again. The default value is set to “43200” (12 hours).

4.1.2 Volumes Configuration

Besides these variables, a few data volumes are necessary for a more convenient use of boi-Das container. These volumes require permissions for **www-data user (uid=33)**:

- `/opt/boidas_backend/media`: This volume is used by the service to keep selfie images, selfie videos, ID document images and the rest of the resources used on a validation. **It should be shared among containers sharing the same database**. This folder requires www-data user permissions and **it must be created before running boi-Das service**.

- `/var/log/boidas`: This volume persists log files generated by boi-Das service. **Each container generates its logs**. *This folder requires www-data user permissions* and **it must be created before running boi-Das service**.

These volumes must be properly configured in the deployment tool scripts, as it is indicated in the section 5.2.1. boi-Das dashboard deployment of the current document, to allow the service docker containers to write and read from them.

As seen in the docker deployment instructions above, there are two volumes that can be mounted in the host machine for convenience.

One is the logs volume where we can search for the log info regarding the boi-Das events.

Several log files are generated as a result of boi-Das operation. The files and the paths where they are generated are the following:

- */var/log/boidas/boidas.access.log*: boi-Das access logs.
- */var/log/boidas/boidas.log*: boi-Das output logs.
- */var/log/boidas/boidasPolling.log*: boi-Das polling logs
- */var/log/boidas/boidasPollingVideocall.log*: boi-Das videocall polling logs

The other one is the media folder which is required for boi-Das data persistence. This volume stores all the media data gathered from user validations, as document photos, image

crops, selfie photos, videos and other resources which were used on vali-Das validations. The folders contained inside this “media data” volume are the following.

Although these media resources can be retrieved by using the boi-Das API and also viewed in the boi-Das GUI, the users of boi-Das could have the necessity of working with these files directly. For this reason, it is convenient for them to know the naming conventions for these resources, allowing them to develop custom scripts and applications for doing ad-hoc processes.

4.1.2.1 validation_export

Contains temporarily the ZIP files created when a validation or a group of validations are exported from the boi-Das dashboard.

4.1.2.2 validation_data

boi-Das media data is stored in a folder structure hierarchy which gives context about the creation date and allows an easier search and organization.

The media folder structure is the following one.

- media
  - validation_data
    - YYYY
      - MM
        - DD
          - VALIDATION_ID

With this folder structure, all the media resources (images, videos, xml, etc.) regarding the validation with the id equals to VALIDATION_ID, are stored under VALIDATION_ID directory,

The naming conventions of the media resources are the following.

“ImageType_ValidationId_TOKEN(_ImageSubtype).Extension”. The name of the video is build as follows: “Video_ValidationId_HASH.Extension”. The image and video extensions are the same than the images and video sent to the server on the requests with the exception of the cases indicated in the section below. The values that can take each part of the image name are the following.

- **ImageType**:
  - obverse
  - obverseFlash
  - reverse

  - selfie
  - selfie_alive
  - nfc_face
  - video
  - videocall

- **ValidationId example**: 6cfb586f4cx343ec205ddf5e28638863

- **Token example:** 116b8c4f33d3171febe2

- **ImageSubtype:**
  - cut: image with the cut of the I.D. document which appears in the photo sent to the server for being analysed. Its file extension is always “jpg”.
  - cut_face: image with the cut of the face which is obtained from the I.D. document photo sent to the server. Its extension is always “png”.
  - cut_signature: image with the cut of the signature which is obtained from the I.D. document. Its extension is always “png”.
  - cut_fingerprint: image with the cut of the fingerprint which is obtained from the I.D. document. Its extension is always “png”.

- **Image and video names examples:**
  - reverse_21ed9fh055ah425g90b5v6c14c2b5854_21gaff7dd302ef2b2fc1_cut.jpg
  - nfc_face_819d9fb085ae4256s0b526carcbbg854_1g7ddfc265fd8b7ab2a0.png
  - obverse_21e69f6055ae4d569sb5edca44bb5454_49g259dbe3d7489b9bbc_cut_signature.png
  - video_63c563896e424dddae470ae24ce4s9s1_cd2er6b4512311d79f3f.mp4
  - videocall_63c563896e424dddae470ae24ce4s9s1_cd2er6b4512311d79f3f.mp4

4.1.3 Video call flow Configuration

When video call flow is chosen, it is necessary to configure the next variables:

VIDEOCALL_URL: ie. https://videocallmanager.eu.veri-das.com=> This URL has to be set to the VideocallManager URL which is deployed on the Veridas SaaS. If this is not correctly set, boi-Das will not be able to do video calls and download video call recordings.

- VIDEOCALL_CLIENTID: “client_id” => This has to be set to the oauth credentials provided by Veridas when VideocallManager documentation was given.
- VIDEOCALL_CLIENTSECRET: “client_secret” => This has to be set to the oauth credentials provided by Veridas when VideocallManager documentation was given.

4.1.4 Second-factor authentication Configuration

By default, boi-Das service authentication is based on user and password verification.
In addition, it is possible to configure “second-factor” authentication functionality that provides more security to the service.

The following variables are necessary when “second-factor” authentication feature is required:

- OTP_ENABLED: “yes” or “no” => If this variable is set to “yes”, the Google Authenticator 2FA is activated for the service instance, requiring all the users to activate and use it. Otherwise, this 2FA is not activated.
- OTP_PERIOD_ENABLED: “yes” or “no” => If this variable is set to “yes”, the 2FA will not be asked to the users during a period of time after the last login. This period of time can be configured by using the env. var. “OTP_PERIOD_HOURS”, and its default value is 24 hours. Default value is set to “no”.
- OTP_PERIOD_HOURS: “24” => how many hours will need to pass until the next OTP authentication requirement. Default value is set to “24” hours.
- GOOGLE_AUTH_ISSUER: “Veridas-Boidas” => There are two fields which identify the 2FA provider in the Google Authenticator QR code. These fields are called "issuer" and "user ID". These values are displayed by the Google Authenticator app just above the 6-digit OTP code. To customize the "issuer" field, this environment variable can be defined. If this variable is not defined, it will default to "Veridas-Boidas". The "user ID" field will always be filled with the username of the user for whom the QR code was created.

4.2 NGINX docker

4.2.1 General Configuration

Apart from the boi-Das container, an NGINX proxy has to be deployed alongside the dashboard. It is used as a statics server, and is also the boi-Das gateway, so the NGINX container host and the port exposed by NGINX container has to be used for accessing boi-Das.

The boi-Das Docker image is prepared for deploying the NGINX server configured and ready to work. Like in the boi-Das Web App container, NGINX container creation procedure is configured by means of some environment variables which can be given to the *docker run* command or in a docker-compose.yml file. The default value of these variables is given after equals sign.

- SERVER_NAME: localhost => The name of the NGINX server used just for internal configuration purposes, so a name like “localhost” is valid.
- NGINX_UPSTREAM: boi-Das Host (i.e. boi-Das container name).
- PORT: 8850 => the port that boidas has configured, 8850 by default.

Also, it is important to note that the ports which NGINX server expose are the 8080 (for HTTP) and the 8443 (for HTTPS). At least one of these ports must be binded to a host port to allow accessing boi-Das service through the NGINX. This binding can be done as it is shown in the docker run and docker-compose examples below.

4.2.2 Volumes Configuration for SSL implementation

Besides these variables, a data volume is necessary for a more convenient use of NGINX container. Some of these containers require permissions for **www-data user (uid=33)**:

- `/etc/boidas/security/certs/`: This volume is necessary for the deployment of boi-Das using SSL (ENABLE_SSL=TRUE.). boi-Das service includes default certificates valid for testing and sandboxing purposes, but for production ready environments, these default certificates must be replaced for valid ones issued by a trusted CA. *This folder requires www-data user permissions* and **it must be created before running boi-Das service**. We recommend mounting this volume with read-only permissions. It should contain the following SSL keys and certs:
  - server.crt
  - server.key

This volume must be properly configured in the deployment tool scripts, as it is indicated in the section 5.2.1. boi-Das dashboard deployment of the current document, to allow the service docker containers to write and read from them.

4.2.3 Deploy on a subdirectory Configuration

By default, boi-Das service is deployed under *root* path like https://my-domain/.

However, it may be necessary to deploy it under a subdirectory or different base path. To do that, the following configuration is required:

- LOCATION: /subdirectory => This variable allows you to configure the subdirectory used at deploy time. The boi-Das will serve requests under the subdirectory indicated by LOCATION (e.g. /boidas).

To complete this configuration, **it is mandatory to modify the BASE_URL variable which is configured in boi-Das docker** (section 4.1.1). The value of BASE_URL must be consistent with the value of the LOCATION var.

- BASE_URL => The URL conformed with the protocol + hostname + port + subdirectory, https://HOSTNAME:PORT/LOCATION (e.g. https://my-domain/boidas).

5. Docker Container Creation

Boidas On Premises comprises the containers deployed by using the given *boidas:BOIDAS-SERVICE_VERSION* Docker image of boi-Das, where current version of the image is the following:

BOIDAS-SERVICE_VERSION = 1.12.5

5.1. boi-Das service deployment

In case you want to use docker tool (https://docs.docker.com/get-started/part2/) for the containers deployment, using a database deployed somewhere else, you can use the following instructions on the terminal. It assumes that the database has been initialized properly with access granted to DB_USER with DB_PASS. It is important to take into account that the services have to be deployed in the same docker network to allow each container to reach each other.

The deployment can also be done by using docker-compose tool (https://docs.docker.com/compose).

We recommend using docker-compose tool because the integration should be easier and only need one step to up boidas service.

For a complete containerized solution which uses a PSQL docker container as a database system, the following steps must be followed.

5.2.1. boi-Das dashboard deployment

If boi-Das is being deployed for the first time, the container and the database have to be initialized. Similarly, if it is being upgraded from a previous version, the database has to be upgraded also before deploying the new version.

To achieve this action, the MIGRATIONS environmental variable should be “yes”.

Please, ensure the VALIDAS_URL and API_KEY are the ones provided by Veridas. The ones shown above are examples.

During boidas-service container start-up, an administrator superuser (with username “admin”) is created and the password for this user is printed on the screen log like the following example (this is just a format example, not a valid password):

```
boidas-service    | BOIDAS SUPERUSER PASSWORD: TexQ2N20xh1zQt3RaOwGvob3NKdEdGi0mR4pw3c1
```

boi-Das service should be exposed at https://localhost:8443/.
In addition, the superadmin section (for DB management) is also reachable at: https://localhost:8443/admin. From here, the admin user password can be changed. Also additional supervisors and/or agent users can be created (this action is also possible via API, View section 7)

```
#docker-compose.yml
version: '2.1'
services:
  pgsql-boidas:
    image: postgres:12.6
    container_name: pgsql-boidas
    environment:
      POSTGRES_DB: boidas
      POSTGRES_USER: boidas
      POSTGRES_PASSWORD: boidas
      PGDATA: /var/lib/postgresql/data/pgdata
    volumes:
      - ./vols/pgdata:/var/lib/postgresql/data/pgdata
  boidas-service:
    image: "${DOCKER_REGISTRY_IP:PORT}/boidas:${BOIDAS-SERVICE_VERSION}"
    container_name: boidas-service
    restart: always
    depends_on:
      - pgsql-boidas
    environment:
      TZ: Europe/Madrid
      DB_HOST: pgsql-boidas
      DB_USER: boidas
      DB_PASS: boidas
      DB_NAME: boidas
      WORKERS: 1
      ENABLE_SSL: "TRUE"
      MIGRATIONS: "yes"
      VALIDAS_URL: ${VALIDAS_URL:-https://api.eu.veri-das.com/validas/v1}
      API_KEY: ${API_KEY:-Gmniefmdslgje7395ngmnweGEke73G}
      BASE_URL: 'https://localhost:8443'
      EMAIL_HOST_NAME: 'smtp.mycompany.com'
      EMAIL_HOST_PORT: '587'
      EMAIL_HOST_USER: "noreply@reply.com"
      EMAIL_HOST_PASSWORD: "1341"
      DEFAULT_FROM_EMAIL: "info-noreply@veridas.com"
   volumes:
      - ./vols/media:/opt/boidas_backend/media
      - ./vols/logs:/var/log/boidas
  nginx:
    image: "${DOCKER_REGISTRY_IP:PORT}/boidas:${BOIDAS-SERVICE_VERSION}"
    environment:
      SERVER_NAME: localhost
```

```
    NGINX_UPSTREAM: boidas-service
    PORT: 8850
  command: /opt/boidas_backend/run_nginx.sh
  depends_on:
    - boidas-service
  ports:
    - 8443:8443
  volumes:
    - ./vols/certs:/etc/boidas/security/certs/
```

For running this docker-compose script, the following command has to be executed, supposing that the docker-compose file shown above is called “docker-compose.yml”.

```
$ docker-compose pull # Just if the images are in a docker registry
$ docker-compose -f docker-compose.yml up --abort-on-container-exit  --force-recreate
```

5.2. Container Cleaning

In case of failure during these steps, the first time it is executed, it is required to clean all volumes and recreate them as empty volumes. Similarly, it is required to remove any container left on the machine. It can be done as follows:

```
$ sudo docker rm CONTAINER_ID
```

6. Additional Considerations

6.1. Boi-Das Inner Decisions Logic

Although boi-Das downloads the vali-Das service confirmed validations as they are, there are certain decisions which are made by boi-Das based on the information contained in these validations.

6.1.1 Validation state based decisions

Once the correctness of a validation has been assured by boi-Das, the validation is downloaded and is assigned to one of the validation state queues based on the following elements:

1. If there is a contextual data with the key “*stats_onboarding_state*” and the value “*rejected*” the validation is considered as rejected, and will appear in the rejected section in the boi-Das GUI.
2. If there is a contextual data with the key “*stats_onboarding_state*” and the value “*approved*” the validation is considered as approved, and will appear in the approved section in the boi-Das GUI.

3. If there is a contextual data with the key "*stats_onboarding_state*" and the value "*approved*", and also there is a contextual data with the key "*stats_videocall*" an the value *true*, the validation is considered as ready to do the video call process, so "pre-approved", and will appear in the pre-approved" section in the boi-Das GUI.
4. If there is no contextual data with the key "*stats_onboarding_state"* the validation is considered as *pending*, and will appear in the pending section in the boi-Das GUI.

6.2 Migrate Postgresql database

If a PSQL docker container is used as a database system, the following steps must be followed if you want to migrate to a new Postgresql version:

1. Keep `pgsql-boidas` container awake and stop `boidas-service` container:

   ```
   docker stop boidas-service
   ```

2. Run the following command:

   ```
   docker exec -it pgsql-boidas pg_dumpall -U boidas > pgdump.out
   ```

3. Stop postgresql container:

   ```
   docker stop pgsql-boidas
   ```

4. Delete postgresql volumes (vols directory) or create a new one (this option is more safe) and assign the corresponding permissions (more information here)
5. Change to desired postgresql (`pgsql-boidas`) image version in docker-compose.yml file.
6. Change `boidas-service` volumes in docker-compose.yml file (if you did create new ones).
7. Run only `pgsql-boidas` container:

   ```
   docker-compose up -d pgsql-boidas
   ```

8. Run the following command:

   ```
   docker cp pgdump.out pgsql-boidas:/
   ```

9. Run the following command:

   ```
   docker exec -it pgsql-boidas psql -f pgdump.out -U boidas
   ```

10. Stop postgresql container:

    ```
    docker stop pgsql-boidas
    ```

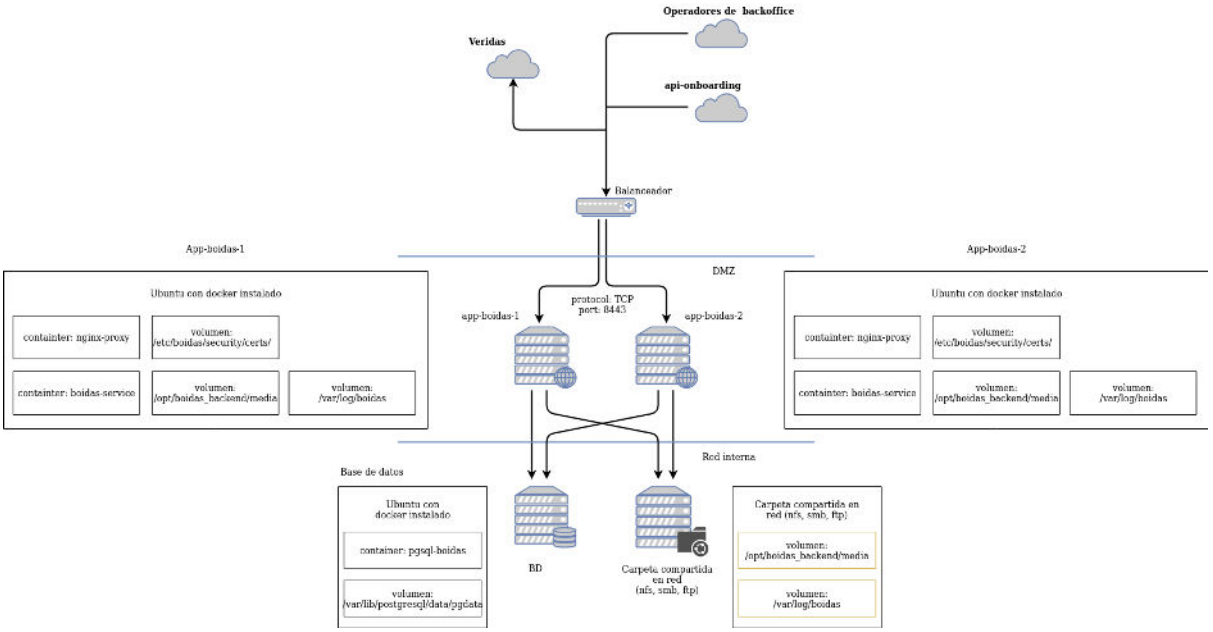
11. Run `boidas-service` deployment as usual

For more information you can check https://www.postgresql.org/docs/10/upgrading.html

6.3 Deploy boi-Das on a High Availability (HA) environment

boi-Das service supports its deployment on a high availability environment, and with all its instances running active-active, acting as a cluster, enabling an horizontal scalability of the service as the load grows at the time the service keeps fault-tolerant. For most users, a two-node cluster will be sufficient for this purpose.

The following diagram depicts the HA architecture:



In order to conduct this deployment, you will consider the following assumptions:

- The container must be configured through the available mechanisms (environment variables) without overriding anything via volumes, new layers, changing the ENTRYPOINT/CMD etc.

- At least 2 instances of nginx+boidas will be deployed. Each nginx will proxy requests to their corresponding api (NOT load balance calls to the API)

- All boidas instances will share a single PostgreSQL database

- All boidas instances will share a volume mount point for media assets (images, etc.) typically using NFS on a linux environment, or EFS on AWS deployments

- Each instance will mount a dedicated volume for logs, or map different folders of the same volume to each boidas container

- boi-Das exposes a HTTPS service under a TCP port for both the UI and the API services (see 4.2 NGINX docker). API calls are stateless, and UI calls rely on a token cached on the user's browser for session persistence. Both services are served by the NGINX service and for HA purposes, the load balancer does not require to use sticky sessions, so a round-robin or leastConnections balancing mechanism should work.
- All instances will run their own polling processes (as is the default on the docker image) with 2 threads each (4 total)

6.4 Product Monitoring and Control

Monitoring is important to determine if the services are working correctly and to early detect potential issues.

Apart from logging files, you could prepare scripts or use monitoring tools to automate the control.

boi-Das service is composed with different parts that it is recommended to check.

6.4.1 Check availability of API service

If you want to check if the service is up, you can create a script using this shell command and verify that the result is 204.

```
curl -I --location --request GET 'https://<base_url>/api/v1/alive'
HTTP/1.1 204 No Content
```

Also, to check if the service connects to the database properly, you will use this other command with a 200 response.

```
curl -I --location --request POST 'https://<base_url>/api/v1/oauth/token/' \
     --header 'Authorization: your credentials oauth
     --form 'grant_type="password"' \
     --form 'username="your username here"' \
     --form 'password="your password here"'
HTTP/1.1 200 OK
```

6.4.2 Check availability of User Interface

Another very important component is the User Interface that you can check if it is running by doing this command. The result of this must be 200.

```
curl -I --location --request GET 'https://<base_url>'
HTTP/1.1 200 OK
```

6.4.3 Check polling for validations service

This service is responsible for connecting to VeriSaaS and downloading the validations availables.

On the one hand, it is necessary to verify that the polling runs every X seconds, by default 60. There is a log file called “boidasPolling” that registers an event each time the polling is launched. You can check it by looking for this event in “boidasPolling.log” file which should be appears every X seconds:

```
{"event": "Running job \"get_validations (trigger: interval[0:01:00], next run at:
2021-08-03 10:45:56 CEST)\" (scheduled at 2021-08-03 10:44:56.237152+02:00)"...}
```

On the other hand, it is very important to check if the polling process connects correctly to VeriSaaS. It may be that the polling works correctly but the connection or download fails. We recommend configuring an alert that is raised when this event appear in “boidas.log” file:

```
{"event": "Problems to obtain validations from Validas”... }
```

6.5 Backup

The following steps must be followed if you want to do a consistent backup of all the boi-Das information:

1. Stop boi-Das service by stopping its docker, or all the dockers of the boidas cluster in case of a distributed implementation

   ```
   docker stop boidas-service
   ```

2. Perform a backup of the postgreSQL database. For more information about Postgresql Backup and Restore options you can check https://www.postgresql.org/docs/9.1/backup.html

3. Backup the volume mount points for media assets and logs. See 4.1.2 Volumes Configuration for more information

4. Start again boi-Das docker, or all the dockers of the boidas cluster in case of a distributed implementation

If the volume mount points are sitting on a storage cabinet you might want to leverage the snapshot capabilities of the cabinet to reduce the downtime due to the backup window. Same applies for the database datastores.

It is highly recommended to perform this backup operation during low user activity to minimize the amount of processes non gathered from the cloud after it resumes its operation.

Important: For security reasons, Veridas has an autocleaner process continuously running at VeriSaas to remove every validation process left at the cloud after a specific time regardless if they were or were not downloaded into boidas. By doing this, we avoid long term persistence of sensible data in the cloud. To avoid losing processes, you should carefully measure the backup process downtime of your backup process so that it is ensured that this time is lower than the autocleaner time period. The autocleaner time can be adjusted by Veridas Customer Support by simply raising a ticket specifying the appropriate time.

6.6 Restore

The following steps must be followed for restoring a backup of all the boi-Das information:

1. Prepare a new deployment following 5.2.1. boi-Das dashboard deployment steps

2. Restore the data of the volume mount points for media assets and logs in the new path and configure the new volume mount point correctly.

3. Run only `pgsql-boidas` container:

   ```
   docker run -d pgsql-boidas
   ```

4. Perform a restore of the postgreSQL database. For more information about Postgresql Backup and Restore options you can check https://www.postgresql.org/docs/9.1/backup.html

5. Stop postgresql container:

   ```
   docker stop pgsql-boidas
   ```

6. Run `boidas-service` deployment as usual

6.7 Data Export

Under some circumstances, the administrator of the system may need to export or migrate all or part of the data stored in boidas.

This task can be easily performed using the export feature, via UI or API.

The selected validations will be exported as a single ZIP file, containing individual folders per each validation process named with their validation IDs. Inside the folder, a media folder will contain all the images, video pieces of evidence as they were originally uploaded to Verisaas and their respective cuts. Also a json file is included containing all the validation data (ocr, scores, etc).

This way an administrator can easily migrate/move the validation processes to another storage/archival system.

Please have a look at section 7.2.2. Get validation by validation_id for further information

7. API

7.1. API Considerations

- The `multipart/form-data` content type must be used on every request.
- The API is HTTP-based and uses SSL with valid certificates.
- Endpoints attempt to conform to the design principles of Representational State Transfer (REST).
- The service includes a GET `/alive` endpoint that returns the 204 HTTP status code if the service is up and running. This can be used to check the service's health.
- The service has an OAuth layer that manages the credentials to use the API. This layer is implemented by the Authorization header (Authorization: Bearer TOKEN). The Application is initialized with a token which does not expire, associated to the user "admin" which is also created by default. This token, as well as the "admin" password and the rest of the credentials needed to use the API (oauth application client_id and client_secret) are:
    - onprem installations: displayed on the screen along the rest log lines during the docker containers initialization stage
    - cloud instance: provided at the time of delivering the service.

All responses are encoded using JSON, regardless of the accepted content-type specified by the client. Responses return a suitable HTTP status code indicating if the request is successful (20X) or not (any other code). Error responses include a message and a code field in the JSON body that provide more information about the concrete error on each case.

In general, successful responses have the following format:

HTTP Status: 200 OK

```
{
      “data”: {
            DATA
      }
}
```

or

HTTP Status: 204 NO CONTENT

where:

Field	Required	Description
data	no	Message payload. It will contain an arbitrary JSON content defined by each endpoint.

In case of API error:

Field	Required	Description
code	yes	Error code
message	no	A message indicating what went wrong
errors	no	A list of the errors providing more information.

In case of OAuth authentication error:

Field	Required	Description
detail	yes	A message indicating what went wrong

7.1.1. Versioning

The API version will be included in the URL, after the base url and before the endpoint:

`https://<base_url>/<service>/v{number:integer}/<endpoint>`

Non backwards compatible changes will cause a version increment.

7.2. API Definition

This service is a REST API that revolves around managing the “validation” resource and it’s subresources, “document”, “selfie”, “video”, “nfc”. The following endpoints are exposed:

Public Base URL (v1)

`https://<base_url>/api/v1/`

Resources

Method	Public URL	Description
POST	`/user`	Creates a user which could request for oauth2 tokens
DELETE	`/user`	Delete a user.
PUT	`/user/role`	Change a user’s role.
PUT	`/user/otp/reset`	Reset a user's OTP.
POST	`/oauth/token`	Retrieves a oauth2 token which expires after 10 days
GET	`/alive`	Checks if the service is up.
GET	`/validation?status=<status>&from=<date>&to=<date>`	Gets a list of validations. Can be filtered by status and/or by date
GET	`/validation/{id}`	Gets a validation
GET	`/validation/{id}/ocr`	Gets validation OCR data obtained from the ID document
PUT	`/validation/{id}/ocr`	Updates the OCR data of the validation, previously obtained using GET /validation/{id}/ocr

GET	/validation/{id}/scores	Gets validation scores (this may be included in the GET validation method)
GET	/validation/{id}/document/image/{image_type}	Gets the images of the document associated to a validation. Image types are: obverse, obserse/cut, reverse, reverse/cut, obverse-flash, obverse-flash/cut, nfc/face, face, etc..
GET	/validation/{id}/selfie	Gets the selfie associated to a validation
GET	/validation/{id}/selfie-alive	Gets the selfie associated to a validation
GET	/validation/{id}/video	Gets the video associated to a validation
GET	/validation/{id}/videocall	Get the video call recording associated to a validation
GET	/validation/{id}/activity	Get the activity registry from a validation
GET	/validation/{id}/timestamp	Get the timestamp zip associated to a validation
PUT	/validation/{id}/acceptance	Accepts a validation (changes the validation state to accepted)
PUT	/validation/{id}/rejection	Rejects a validation (changes the validation state to rejected)
PUT	/validation/{id}/inconclusive	Inconclusive validation (changes the validation state to inconclusive)
PUT	/validation/{id}/preacceptance	Preaccepts a validation (changes the validation state to preaccepted)
DELETE	/validation/{id}	Deletes a validation
POST	/contextual/filter	Creates a contextual data filter
PUT	/contextual/filter/{filter_id}	Modify a contextual data filter
GET	/contextual/filter	Get all the contextual data filters
GET	/contextual/filter/{filter_id}	Get specific contextual data filter
DELETE	/contextual/filter/{filter_id}	Delete contextual data filter

7.2.1. Check if the service is alive

`GET /alive`

Response example

HTTP Status: 204 NO CONTENT

7.2.2. Get validation by validation_id

Retrieves a validation.

`GET /validation/{validation_id}`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN
(Accept-boidas: zip)

If the optional header "Accept-boidas" is configured with the value "zip", the validation data is returned in a ZIP format.

Response example 1 - JSON

HTTP Status: 200 OK

```
{
  "data":
    {
      "createdAt": DATETIME,
      "data": {
       "videoCall": {
          "state": STATE,
          "client_date": DATE,
          "started_videocall_at": DATE,
          "ended_videocall_at": DATE,
          "has_recording": TRUE/FALSE,
          "href_recording: URL
       },
       "biometry": {
          "_links": {...},
          "scores": [...]
       },
       "integrity": {...},
       "summary": {
          "scores": [...],
          "notValidatedScores": [...]
```

```
        },
        "documentVsVideo": {...},
       "challenges": {...},
        "contextualData": [...],
        "comments" : [
           {
             "comment": COMMENT,
            "user": USER,
            "created_at": DATE
           }
        ]
        "document": {
          "_links": {...},
          "nodes": [...],
         "scores": [...]
        },
        "identidasVersion": IDENTIDAS_VERSION,
        "timestamp": {...},
      },
      "documentId": DOC_ID,
      "documentType": DOCTYPE,
      "id": VALIDATION_ID,
      "pollAnswers": {...},
      "status": VALIDATION_STATUS
    }
}
```

Response example 2 - ZIP

Binary corresponding to the ZIP file.

A header named “Accept-Boidas” with the value “zip” indicates that this type of response is wanted.

This ZIP will contain 2 JSON files. In the user_data.json, a field named "comments" includes the comments given by the agent to the validation through the boidas UI.

7.2.3. Get validations

7.2.3.1. Get validations By status, from and to date

Retrieves validations:

GET
`/validation?status=<status>&from=<date>&to=<date>&statusChangedBefore=<datetime>&statusChangedAfter=<datetime>`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

`Where status can be: confirmed(pending), inconclusive, preapproved, accepted, rejected, date follows YYYYMMDD format and datetime follows YYYY-MM-DDTHH:MM:SS`

It is important to take into account that createdAt validation date has the following format: GMT + CET/CEST.

This endpoint has few basic functionalities for pagination and ordering through these parameters:

- **page**: Current page
- **perPage**: number of items per page (to a maximum of 100)

Examples

GET /validation?status=<status>&page=1&perPage=5

Response example

HTTP Status: 200 OK


```
{
 [ {
    "data":
    {
      "createdAt": "2018-03-16 12:50:47.918027",
       "updatedAt": "2018-03-16 12:54:47.918027",
       “stateChangedAt”: “2018-03-16 13:50:00.918027
      "data": {
        "biometry": {},
        "contextualData": [],
        "document": {
          "images": {},
          "nodes": {},
          "scores": {}
        },
        “integrity”: {...},
        “summary”: {
            “scores”: [...],
            “notValidatedScores”: [...]
         },
        “documentVsVideo”: {...},
```

```
        “challenges”: {...},
         "identidasVersion": 1.14.3,
         “timestamp”: {...}
       },
       "documentId": null,
       "documentType": null,
       "id": "7e84ff4eb000409892c9995e4955e636",
       "pollAnswers": null,
       "state": "0"
     }
}
},
 "data":
     {
       "createdAt": "Mon, 30 Apr 2018 09:52:02 GMT",
       "data": {
         "biometry": {},
         "contextualData": [],
         "document": {
           "images": {},
           "nodes": {},
           "scores": {}
         },
         "identidasVersion": 1.14.3
       },
       "documentId": null,
       "documentType": null,
       "id": "7e84ff4eb000409892c9995e4955e636",
       "pollAnswers": null,
       "state": "0"
     }
}
}
...
]
}
```


Errors:

Code	HTTP Status	Message
InvalidParameter	400	Url parameters not valid
InvalidDateFormat	400	Invalid date format. The correct format is YYYYMMDD

InvalidDatetimeFormat	400	Invalid datetime format. The correct format is YYYY-MM-DDTHH:MM:SS
UnknownStateError	400	Unknown validation state

7.2.3.2. Get just certain fields of the validations

There is the possibility of just retrieving certain fields of the validations by using the query string parameter called “include”.

```
GET /validation?include=id,scores,nodes,createdAt,state
```

Content-type: multipart/form-data

Examples

```
GET /validation?include=id
GET /validation?include=createdAt,state
```

7.2.4. Get OCR

Retrieves the document ocr originally obtained by the Document Analysis service.

```
GET /validation/{validation_id}/ocr
```

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Response example

HTTP Status: 200 OK


```
{
  "data": {
    "nodes": [
      {
        "fieldName": "Nombre / Name",
        "name": "PD_Name_Out",
        "text": "JUAN"
      },
      {
        "fieldName": "Apellidos / Last Names",
        "name": "PD_LastName_Out",
```

```
    "text": "ESPA\u00d1OL ESPA\u00d1OL"
  },
  {
    "fieldName": "DNI / DNI",
    "name": "PD_IdentificationNumber_Out",
    "text": "00000051T"
  },
  {
    "fieldName": "Fecha de Validez / Expiration Date",
    "name": "DD_ExpirationDate_Out",
    "text": "01 06 2016"
  },
  {
    "fieldName": "IDESP / IDESP",
    "name": "DD_DocumentNumber_Out",
    "text": "AAA000000"
  },
  {
    "fieldName": "Sexo / Gender",
    "name": "PD_Sex_Out",
    "text": "M"
  },
  {
    "fieldName": "Nacionalidad / Nationality",
    "name": "PD_Nationality_Out",
    "text": "ESP"
  },
  {
    "fieldName": "Fecha de Nacimiento / Date of Birth",
    "name": "PD_BirthDate_Out",
    "text": "01 01 1951"
  },
  {
    "fieldName": "Municipio de Nacimiento / Town of Birth ",
    "name": "PD_BirthPlaceMunicipality_Out",
    "text": "MADRID"
  },
  {
    "fieldName": "Provincia de Nacimiento / Province of Birth",
    "name": "PD_BirthPlaceState_Out",
    "text": "MADRID"
  },
  {
    "fieldName": "Domicilio / Address ",
    "name": "PD_AddressStreet_Out",
    "text": "C JULIAN GONZALEZ SEGADOR S N"
  },
  {
```

```
        "fieldName": "Municipio de Domicilio / Town of Residence",
        "name": "PD_AddressMunicipality_Out",
        "text": "MADRID - MADRID"
      },
      {
        "fieldName": "Provincia de Domicilio / Province of Residence",
        "name": "PD_AddressState_Out",
        "text": "MADRID"
      }
    ]
  }
}
```


7.2.5. Get validation scores

Retrieves the scores obtained in document and biometry analysis.

`GET /validation/{validation_id}/scores`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Response example

HTTP Status: 200 OK


```
{
  "data": {
    "biometricScores": [
      {
        "name": "ValidasScorePhotoId",
        "value": 0.04565465917179558
      },
      {
        "name": "ValidasScoreSelfie",
        "value": 0.04565465917179558
      },
      {
        "name": "ValidasScoreVideo",
        "value": 0.09144748463204097
      },
    ],


    "documentScores": [
      {
        "name": "ValiDasMRZPaisExpedicionValue1",
```

```
        "value": 1
      },
      {
        "name": "ScoreHijoDeOCR",
        "value": 1
      },
      {
        "name": "ValiDasMRZFechaDeValidezRegular1",
        "value": 1
      },
     ...
    ],
    "integrityScores": [
      {
        "name": "ValidasScoreIntegrity",
        "value": 1
      },
    ],
    "summaryScores": [
      {
        "name": "ValidationGlobalScore",
        "value": 0.0
      },
    ],
    "notValidatedScores": [
      {
        "name": "ValidasScoreSelfie",
        "Value": 0.04565465917179558
      },
      {
        "name": "ValidasScoreVideo",
        "value": 0.09144748463204097
      },
    ],
    "documentVsVideoScores": [
      {
        "name": "ValidasScoresVideoVSDocument",
        "value": 1.0
      },
    ],
  }
}
```


7.2.6. Get ID document images

Retrieves the ID document related images.

```
GET /validation/{id}/document/image/{image_type}
```

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Where image_type can be on of the following:

- obverse
- obverse/cut
- reverse
- reverse/cut
- obverse-flash
- obverse-flash/cut
- nfc/face
- face
- signature
- fingerprint

7.2.7. Get ID selfie image

Retrieves the selfie image.

`GET /validation/{id}/selfie`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

7.2.8. Get ID selfie alive image

Retrieves the selfie alive image.

`GET /validation/{id}/selfie-alive`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

7.2.9. Get ID selfie video

Retrieves the selfie video.

`GET /validation/{id}/video`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

7.2.11. Put OCR

Updates the document ocr.

`PUT /validation/{validation_id}/ocr`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Name	Req.	Type	Description
correctedOCR	yes	string	Strings with the corrected ocr fields

Example

```
[
	{"name": "PD_BirthDate_Out", "confirmedText": "01 01 1951"},
	{"name": "PD_AddressMunicipality_Out", "confirmedText": "MADRID"}
]
```

Response example

HTTP Status: 204 NO CONTENT

Example

```
correctedOCR:
[
	{
	  "name": "PD_Name_Out",
	  "confirmedText": "JUAN"
	},
	{
	  "name": "Apellidos / Last Names",
	  "confirmedText": "ESPA\u00d1OL ESPA\u00d1OL"
	},
	{
	  "name": "PD_IdentificationNumber_Out",
	  "confirmedText": "00000051T"
	},
	{
	  "name": "DD_ExpirationDate_Out",
	  "confirmedText": "01 06 2016"
	},
	{
	  "name": "DD_DocumentNumber_Out",
```

```
      "confirmedText": "AAA000000"
    },
    {
      "name": "PD_Sex_Out",
      "confirmedText": "M"
    },
    {
      "name": "PD_Nationality_Out",
      "confirmedText": "ESP"
    },
    {
      "name": "PD_BirthDate_Out",
      "confirmedText": "01 01 1951"
    },
    {
      "name": "PD_BirthPlaceMunicipality_Out",
      "confirmedText": "MADRID"
    },
    {
      "name": "PD_BirthPlaceState_Out",
      "confirmedText": "MADRID"
    },
    {
      "name": "PD_AddressStreet_Out",
      "confirmedText": "C JULIAN GONZALEZ SEGADOR S N"
    },
    {
      "name": "PD_AddressMunicipality_Out",
      "confirmedText": "MADRID - MADRID"
    },
    {
      "name": "PD_AddressState_Out",
      "text": "MADRID"
    }
]
```


Errors:

Code	HTTP Status	Message
InvalidData	400	Invalid json data

7.2.12. Accept validation

Updates the validation to accepted (approved) status.

PUT `/validation/{validation_id}/acceptance`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Response example

HTTP Status: 204 NO CONTENT

Errors:

Code	HTTP Status	Message
`VideocallRecordingError`	`400`	`Recording video call does not exist`

7.2.13. Reject validation

Updates the validation to rejected status.

PUT `/validation/{validation_id}/rejection`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Response example

HTTP Status: 204 NO CONTENT

7.2.14. Inconclusive validation

Updates the validation to inconclusive status.

PUT `/validation/{validation_id}/inconclusive`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Response example

HTTP Status: 204 NO CONTENT

7.2.15. Preapproved validation

Updates the validation to preacceptance (preapproved) status.

`PUT /validation/{validation_id}/preacceptance`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Response example

HTTP Status: 204 NO CONTENT

Errors:

Code	HTTP Status	Message
`NotAllowedPreapproved`	`409`	`Pre-approved validation is not allowed. Videocall is not required`

7.2.16. Delete a validation

Deletes a validation and all the related files. Just confirmed and cancelled validations can be deleted.

`DELETE /validation/{validation_id}`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Response example

HTTP Status: 204 NO CONTENT

7.2.17. Get video call recording

Retrieves the video call recording.

`GET /validation/{id}/videocall`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

7.2.18. Register a user

Registers a user which will be able to request OAuth2 tokens with a finite life for accessing the API. These users also have access to the boi-Das GUI with their corresponding role.

`POST /user`

Content-type: multipart/form-data
Authorization: Bearer $SUPERVISOR_OAUTH2_TOKEN

Name	Req.	Type	Description
username	yes	string	Username of the user to be registered
password	yes	string	Password of the user to be registered
role	no	string	Optional. The role for the new user. There are two options: supervisor or agent. The default role will be “agent”.

Response example

HTTP Status: 204 NO CONTENT

Errors:

Code	HTTP Status	Message
`UserAlreadyExistsError`	`400`	`The user already exists`
`BadRequestError`	`400`	`Your request is not valid and could not be processed`

7.2.19 Delete a user

Delete an existing user. All the user’s OAuth2 tokens will be automatically invalidated.

`DELETE /user`

Content-type: multipart/form-data
Authorization: Bearer $SUPERVISOR_OAUTH2_TOKEN

Name	Req.	Type	Description

username	yes	string	Username of the user to be deleted

Response example

HTTP Status: 204 NO CONTENT

Errors

Code	HTTP Status	Message
`NotFoundUser`	`404`	`Resource has not been found`

7.2.20 Change a user's role

Change a user's role.

PUT `/user/role`

Content-type: multipart/form-data
Authorization: Bearer $SUPERVISOR_OAUTH2_TOKEN

Name	Req.	Type	Description
username	yes	string	Username of the user whose role is to be changed
role	yes	string	New role of the user

Response example

HTTP Status: 204 NO CONTENT

Errors

Code	HTTP Status	Message
`NotFoundUser`	`404`	`Resource has not been found`

7.2.21. Reset user's OTP

A supervisor can reset any user's OTP. When a user's OTP is reset, a new QR code will be generated and shown the next time that user logs into boidas.

PUT `/user/otp/reset`

Content-type: multipart/form-data
Authorization: Bearer $SUPERVISOR_OAUTH2_TOKEN

Name	Req.	Type	Description
username	yes	string	Username of the user to reset OTP

Response example

HTTP Status: 204 NO CONTENT

Errors

Code	HTTP Status	Message
`NotFoundUser`	`404`	`Resource has not been found`
`NotAllowed`	`403`	`Only a supervisor can reset a user's OTP`
`NotFoundOTP`	`404`	`OTP has not been found for that user`

7.2.22. Obtain OAuth2 token

Creates and retrieves a token for accessing the API. These tokens expire after 10 days.

`POST /oauth/token`

Content-type: multipart/form-data
Authorization: Basic $credentials

where $credentials is the Base64 encoding of client_id and client_secrect joined by a single colon:

Basic Authentication header has to be sent to authenticate the client application (boiDas) against the oauth2 server deployed within the service. The construction of this Basic Auth header is, as usual, by typing the word Basic plus the base64-encoded username:password, which correspond to client_id:client_secret. For example, the Base64 version of testuser:testpwd is dGVzdHVzZXI6dGVzdHB3ZA== so the Value for the header would be *Basic dGVzdHVzZXI6dGVzdHB3ZA==*

Name	Req.	Type	Description
grant_type	yes	string	Type of the token. Value: **password**
username	yes	string	username of the user that is requesting the token
password	yes	string	Password of the user that is requesting the token

For on premises installations, the configured client_id and client_secret values for the boidasAPI oauth application are displayed on the screen during the docker containers initialization stage alongside the rest of the initialization processes log lines.

```
boidas-service-local    | BOI-DAS API SUPERUSER: admin
boidas-service-local    | BOI-DAS API OAUTH2 CLIENT_ID: uYjiqNPlYZIR3X3VZ6dNPUJi
boidas-service-local    | BOI-DAS API OAUTH2 CLIENT_SECRET: XUClkf4wMdLnLS2RvHo9r
boidas-service-local    | BOI-DAS API SUPERUSER TOKEN: pqWqjvCSwruOiBzR
```

For cloud instances of boi-Das, these credentials are provided together with the rest of the needed information to use the service at the time of the service delivery by the Veridas team.

Response example

HTTP Status: 200 OK

```
{
    "access_token": "BqRADePHLW7s8G9WrwmpErPxMZu4C0rbEF7YNKreaq",
    "expires_in": 864000,
    "token_type": "Bearer"
}
```

Errors

Code	HTTP Status	Message
InvalidClient	401	OAuth Client authentication failed

7.2.23. Get activity registry

Retrieves the activity registry from a given validation.

`GET /validation/{validation_id}/activity`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Response example

HTTP Status: 200 OK

```
{
  "data": [
    {
      "created_at": DATE,
      "user": USER,
      "role": ROLE,
      "action": "Validation accessed via API",
    },
    {
      "created_at": DATE,
      "user": USER,
      "role": ROLE,
      "action": "Validation saved",
    }
}
```

7.2.24. Get timestamp

Retrieves the timestamp zip file from a given validation.

`GET /validation/{validation_id}/timestamp`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Response example

Binary corresponding to the ZIP file.

This ZIP will contain 2 ZIP files, the evidences sent to timetamp and the other with the hash and hash timestamped.

7.2.25. Create contextual data filter

There is an option to segregate validations by user. The goal is that each user can only access validations that have specific contextual data information.

Creates a contextual data filter (key-value) so the selected users will only see the validations whose contextual data is associated to that key-value filter.

The “key” field is unique, meaning that there can only exist one filter with the same key.

Once the contextual data filter has been created, when the user logs in, he will only see those validations whose contextual data contains the filter(key-value) created.

Only users with a supervisor role have access to this endpoint.

`POST /contextual/filter`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Name	Req.	Type	Description
users	yes	string	List of users that will be associated with the filter. If more than one user is sent, the users must be separated by commas. Example: user1, user2, user3, etc.
key	yes	string	Key of the filter
value	yes	string	Value of the filter

Response example:

HTTP Status: 201 CREATED

```
{
  "data": {
    "filter_id": "a1b2",
    "users": [
      "user1",
      "user2"
    "filter": {
      "key": "device",
      "value": "iphone"
    }
  }
}
```

Errors:

Code	HTTP Status	Message
ForbiddenError	403	You do not have permission to perform this action
NotFoundUser	404	Resource has not been found
FilterAlreadyExistError	400	The filter already exists

7.2.26. Modify contextual data filter

Modifies an existing contextual data filter.

The fields that can be updated are the users associated with the filter_id and the value of the filter.

Only users with a supervisor role have access to this endpoint.

PUT `/contextual/filter/{filter_id}`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Name	Req.	Type	Description
users	no	string	List of users that will be associated with the filter. If more than one user is sent, the users must be separated by commas. Example: user1, user2, user3, etc.
value	no	string	Value of the filter

At least one of the parameters must be present in the request.
If the 'users' parameter is present, the previous users associated with the filter_id will be deleted and modified with the ones in the request.

Response example:

HTTP Status: 200 OK

```
{
  "data": {
    "filter_id": "a1b2",
    "users": [
```

```
      "user1"
    "filter": {
      "key": "device",
      "value": "android"
    }
  }
}
```

Errors:

Code	HTTP Status	Message
ForbiddenError	403	You do not have permission to perform this action
NotFoundUser	404	Resource has not been found
FilterAlreadyExistError	400	The filter already exists

7.2.27. Get all contextual data filter

Returns a list with all the information related to all contextual data filters.

Only users with a supervisor role have access to this endpoint.

`GET /contextual/filter`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Response example:

HTTP Status: 200 OK

```
{
  "data": [
    {
      "filter_id": "a1b2",
      "users": [
        "user1"
      "filter": {
        "key": "device",
        "value": "android"
      },
    {
      "filter_id": "b1c2",
```

```
      "users": [
        "user1",
        "user2",
      "filter": {
        "key": "localization",
        "value": "madrid"
      }
    }
  ]
}
```

Errors:

Code	HTTP Status	Message
ForbiddenError	403	You do not have permission to perform this action

7.2.28. Get specific contextual data filter

Returns all the information related to a specific contextual data filter.

Only users with a supervisor role have access to this endpoint.

`GET /contextual/filter/{filter_id}`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Response example:

HTTP Status: 200 OK

```
{
  "data": {
    "filter_id": "a1b2",
    "users": [
      "user1"
    "filter": {
      "key": "device",
      "value": "android"
    }
  }
}
```

Errors:

Code	HTTP Status	Message
ForbiddenError	403	You do not have permission to perform this action

7.2.29. Delete contextual data filter

Deletes an existing contextual data filter.

Only users with a supervisor role have access to this endpoint.

`DELETE /contextual/filter/{filter_id}`

Content-type: multipart/form-data
Authorization: Bearer $OAUTH2_TOKEN

Response example:

HTTP Status: 204 NO CONTENT

Errors:

Code	HTTP Status	Message
ForbiddenError	403	You do not have permission to perform this action

7.3. Generic API errors definition

Errors:

Code	HTTP Status	Message
ValidationError	400	The provided data is not valid.
BadRequestError	400	Your request is not valid and could not be processed
NotFoundValidation	404	Resource has not been found

MethodNotAllowed	405	The requested method is not allowed for this endpoint
IncompleteValidation	409	This action can not be done because the validation is incomplete
NotAllowed	409	Validation state (X) does not allow X
Gone	410	The resource is gone
InternalServerError	500	An error occurred while processing your request

8. Annex 1: License Disclaimer

A license disclaimer can be seen inside the docker container in the path /opt/boidas_backend/license.txt:

The following clauses set the terms, rights, restrictions and obligations on using this Software, created and owned by VERIDAS DIGITAL AUTHENTICATION SOLUTIONS, S.L. (the Licensor), without prejudice to the provisions laid down in the contracts subscribed by the Licensor and your entity (the Licensee), which shall prevail over this file.

LICENSE GRANT
Licensor hereby grants to the Licensee a non-exclusive, non-assignable and non-transferable, indivisible, without the rights to create derivative works license to use this Software for the specific purpose specified between the parties, subject to the terms and conditions contained herein and other legal restrictions set forth in third party software used while running the Software.
The Software has different components that can be used for several applications. However, the License is granted over the Software licensed components as a whole, and no separated use is permitted other than the specific purposes agreed by the Licensor and the Licensee.
The Software is comprised of proprietary code. However, the Software may include certain third party components with separate legal notices or governed by other agreements. Even if such components are governed by other agreements, the disclaimers and the limitations on and exclusions of damages below also apply.
All intellectual and industrial property rights over and in respect of the Software are owned by the Licensor. The Licensee does not acquire any rights of ownership in the Software.
The use of reverse engineering practices is strictly forbidden.

LIMITATION OF LIABILITY AND INDEMNITY

1. To the extent permitted under the law, the Software is provided under an “AS IS” basis. The Licensee acknowledges and agrees that neither the Licensor nor its board members, employees or agents, will be liable for any loss or damage arising out of or resulting from Licensor’s provision of the Software under this License, or any use of the Software by the Licensee or its employees; and Licensee hereby releases Licensor to the fullest extent from any such liability, loss, damage or claim.

Regarding the processing of personal data with the Software, Licensee acknowledges and agrees that Licensor is no liable for the data collected by the use of the Software within the scope of the Licensee’s activities.

2. The Licensee must indemnify, defend and hold harmless the Licensor, its board members, employees and agents from and against any and all claims (including third party claims), demands, actions, suits, expenses (including attorney’s fees) and damages (including indirect or consequential loss) resulting in any way from:

a. Licensee’s and Licensee’s employee’s use or reliance on the Software;

b. any breach of the terms of the License by the Licensee or its employees;

c. any other act of Licensee that can be considered negligent.

3. The facial verification functionality of this Software has been tested against the LFW database, with results up to a 98.5% precision (FNR1.7% at FPR1.5%). This level of reliability is subject to image capture conditions during the process, and therefore the Licensor shall not be responsible of any use the Licensee may make of the Software in different environments than those recommended by the Licensor.

9. Annex 2: Changelog History

boi-Das v1.9.0

General

- **Added**
  - Allow to deploy the service on a subdirectory using LOCATION and BASE_URL env variables (View section 4.2.3)

boi-Das v1.8.0

General

- **Fixed**
  - Configure boi-Das docker service to allow it to run with www-data user (uid=33). Root permissions are no longer required to deploy it

The port that is exposed from NGINX docker has changed. It is mandatory to modify port 443 to 8443 in the deployment step

If using a non-root user to deploy boi-Das, it is mandatory to review and grant the correct permissions to the boi-Das docker volumes if they already existed from previous versions.

```
chown www-data:xxx volume_folder
chmod 775 volume_folder
```

User Interface

- **Added**
  - It is mandatory to add a reason when a validation is rejected. If it does not exist, an error message will be displayed
  - Logout automatically after a certain period of time without activity. The default value is 10 minutes, but it is possible to modify using the new environment variable SECONDS_TIMEOUT_SPA
  - New section is shown in the main header to indicate the global result of the entire process. If the principal scores are valid, it is recommended to accept the process. If not, it is recommended to review it and the causes will be noted in this section as well
  - When a process has a challenge and video evidences, the challenge is visible next to the video

- **Changed**
  - Improve logging registry
  - Change "Image Integrity" score text to "Evidences Integrity"
  - Restyle scores section in MainInformation header
  - Security improvements regarding user roles authorization and application core framework logic separation.

- **Fixed**
  - Show score between document vs video files in Face Biometry tab when the document + video flow is done
  - A validation is now unlocked when a filter is set and the user goes back from a validation to the main dashboard

API

- **Added**
  - New API endpoint which returns the activity registry of a validation
  - Add password validation when create a new user

- **Changed**

- The endpoint GET validation return the validation comments created

- **Fixed**
  - Fix the behaviour of the endpoint GET validation when using the “from” and “to” query parameters

boi-Das v1.6.2

User Interface

- **Added**
  - New “documentVsVideo” score is shown in the main scores section to indicate if the document shown in the selfie video and the document previously analysed are the same or not for Mexican ID Cards. This allows us to comply with the regulatory requirement of the CNBV. Art. 51. Bis 6 Fracción VII -c- (https://dof.gob.mx/nota_detalle.php?codigo=5602349&fecha=12/10/2020)

Export validations

Improved export information data including new “documentVsVideo” scores