das-FaceBond v1.6

Docker Instructions

Revisión	Fecha	Descripción	Redactado	Revisado	Aprobado
1	15/07/2021	v1.6	PZM	DGS	JGC/MSY/PZM

INDEX

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

0. What's new?

This release introduces the following changes:

- Increased identification speed for galleries of up to 10,000,000.
    - The time needed for a high accuracy identification in a 1,000,000 identities gallery has been reduced to less than 1 second.
    - The time needed for an exact identification in a 1,000,000 identities gallery has been reduced to 5.8 seconds.
- Support for das-Face 3.0 or later.
- Update of identification performance when used with das-Face 3.2.
    - Our engine has been evaluated by NIST achieving a 2.78% False Negative Identification Rate (FNIR) on a 640K identities gallery. A 3.73%, 4.91%, and 7.53% FNIR have been reported for 1.3M, 3M, and 6M identities galleries respectively.
    - The new das-Face model improves clustering accuracy from 97.5% to 97.6%.

1. Introduction

In biometrics, there are three main operations:

- Verification: Useful for person identity checking, it consists of comparing two faces, one you know the identity, and another you want to check belongs to the same person. Operations related with facial verification are offered by Veridas into the das-Face product.
- Identification: Used to retrieve the identity of a person, it consists of looking for a particular person (probe) among a large set (called gallery hereinafter), recovering the most confident candidates. By association of the given probe with one candidate, it is possible to identify the probe.
- Clustering: Used to organize a set of data in order to create groups based on the similarity of the members.

The face recognition engine developed by Veridas was ranked by NIST in the top 25% best systems in the world in the MUGSHOT category on April 16th, 2021, and it's the subject of continuous development and improvement efforts.

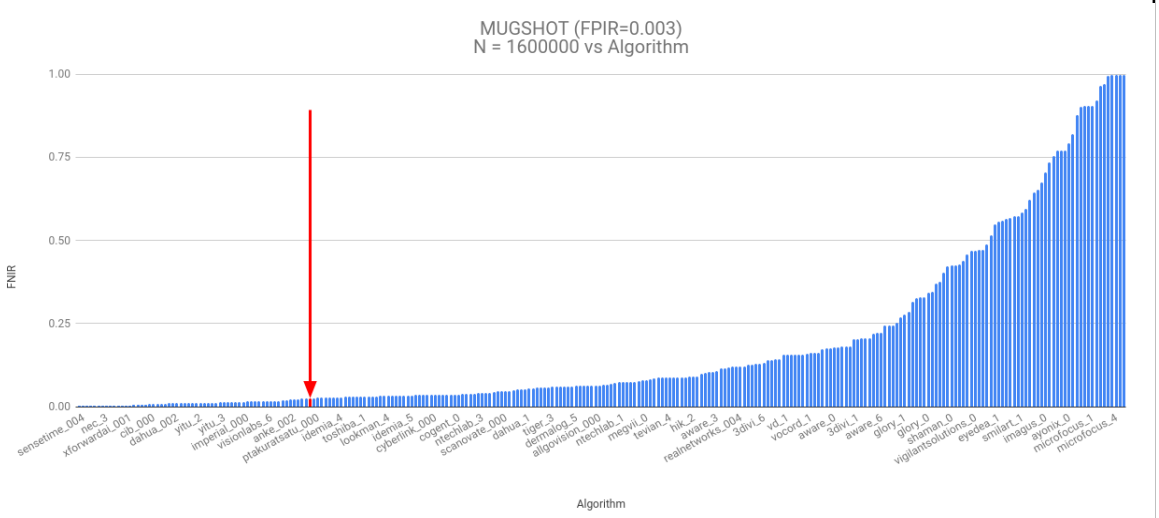
The face recognition engine developed by VERIDAS was ranked by NIST in the 63 of 271 systems presented to FRVT 1:N to the MUGSHOT category. The evaluation was performed on 2021 April.[1] Find below a picture of all the competitors in the mentioned MUGSHOT category. The VERIDAS system has been marked in red

[1] https://pages.nist.gov/frvt/reports/1N/frvt_1N_report.pdf

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

(Results shown from NIST do not constitute an endorsement of any particular system, product, service, or company by NIST.)[2]

VERIDAS achieved a False Negative Identification Rate (FNIR) of 2.44% for a False Positive Identification Rate (FPIR) threshold fixed at 0.3%, and with a gallery with N=1.6M.



The MUGSHOT category is characterized by a collaborative subject almost following ISO 19794-5, so the person whose face is being captured is in good acquisition conditions.

das-FaceBond service has been built to offer identification and clustering operations over galleries of any number of faces. One face corresponds to one image containing a face person and a metadata associated with it. Face comparisons are performed by means of an instance of our das-Face docker image.

Identification operations consist in the comparison of a probe face image with a large previously populated gallery of face images. This operation may be performed in batch, using as probe images a previously populated gallery and as target another gallery. In order to facilitate identifications with large galleries, das-FaceBond offers the role of agent users[3], which are granted with credentials to manually review the operations.

Clustering operations consist in finding natural groups of faces with high similarity, potentially belonging to the same identity. This operation is performed over a previously populated gallery.

[2] https://www.nist.gov/programs-projects/face-recognition-vendor-test-frvt-ongoing
[3] See appendix for more information.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

The way of communicating with this service is following a REST API which accepts bodies with application/json, application/x-www-form-urlencoded, multipart/form-data, image/jpeg or image/png content types. Each endpoint will be documented here indicating which kind of content type is accepted. The response of the service may be an application/json, media/jpeg or media/png content types.

The production version of das-FaceBond service runs on a Docker container which is built on top of Ubuntu 16.04 LTS and that can be installed over an Ubuntu and also a RHEL7.6. Regarding the Docker nature, the containers could be deployed on other Linux distributions. However, the dependencies installation process is just prepared for the two indicated above, so although in certain distributions the installation could also work, there is no guarantee about that.

2. System components

The service das-FaceBond requires a network of different services for running all the features of the system:

- das-Face docker version >= 2.0.0.
  - It is recommended to run it with at least 3 workers with a GPU available. This will require ~5GB per worker.
- Redis docker version >= 4.0.
  - No special configuration is required.
- PostgreSQL docker version >= 9.6.1.
  - It is recommended to use an SSD disk for the database.
- das-FaceBond docker version 1.15.0.
  - It may run with at least 500MB of memory per worker.
- Celery running the image of das-FaceBond.
  - Depending on your parallel requirements, more than one celery instances may be required.
  - The should be executed with at least 1GB of memory per celery worker.
- Nginx running the image of das-FaceBond.
  - It is used as a gateway between Gunicorn process in das-FaceBond and the outside world.
  - It serves static data of das-FaceBond.
  - Required to execute “admin” dashboard of Django.[4]
- das-FaceBondUI docker version 1.0.2.
  - It runs a web UI for exploitation of the system.
- Nginx running the image das-FaceBondUI.

[4] https://docs.djangoproject.com/en/2.2/ref/contrib/admin/

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

- It is used as a gateway between Gunicorn process in das-FaceBondUI and the outside world.
- It serves static data for das-FaceBondUI.

At the end of this document you may find a docker-compose YAML description which puts all these containers up on a single server. Use it as an example for your own deployments.

3. Technical specifications

Currently, the system is limited to work with galleries of up to 10.000.000 faces each.

3.1. Hardware requirements

The minimum requirements for production purposes of a machine running the das-FaceBond system:

- CPU with at least 4 CPU cores (real, not virtual), at least 2.3GHz and 25M of cache.
- GPU 1080 Ti with 12 GB of memory, or superior.
- At least 21GB of RAM memory.
- At least a disk with 40GB of space for 1.000.000 images in a gallery. Each gallery with 1M images supposes additional 40GB of space. These numbers depends on the images of the client. We take a mean estimation based on images of MegaFace benchmark.[5]
- Recommended 10Gbps Ethernet connection.

The database will require a PostgreSQL running in a “db.m5.xlarge” Amazon AWS instance:

- CPU with 4 cores of 2.3GHz and 25M of cache.
- At least 8GB of RAM memory for each gallery of 1M images, requiring additional 8GB per each additional 1M images gallery.
- SSD storage for high performance, with at least 120GB of free space for a gallery of 1M images, adding 120GB additional space per each additional gallery with 1M images. It is recommended an SSD unit with 300GB in order to accomodate scaling over time.
- Recommended 10Gbps Ethernet connection.

Both machines are recommended to be in a dedicated network, in order to reduce latency and to ensure high bandwidth. Database may be deployed on the same machine as das-FaceBond, but if it is the case, the machine should be escalated to have the sum of the requirements of both machines (memory, CPU, storage, …), and we don’t recommend it.

[5] http://megaface.cs.washington.edu/

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

3.2. Accuracy and performance

Main operations of das-FaceBond (clustering and identification) are configurable by a given accuracy option. Such an option indicates how the system will proceed, and has the following accuracy values:

- EXACT: Requests to perform the operation exactly, being the more accurate option, it may be unfeasible for huge datasets (for instance, identifications over thousands of faces, or clustering over hundreds of faces) because of the time required to finish the operation.
- HIGH: Reduces the accuracy of the system, but allows to perform identification over hundreds of thousands of faces, and clustering with thousands of faces.
- MEDIUM: Reduces the accuracy of the system, but allows clustering with dozens of thousands of faces.
- LOW: The less accurate one, but allowing clustering on hundreds of thousands of faces.

This system has been evaluated on an Intel(R) Core(TM) i9-7900X CPU @ 3.30GHz, with 64GB of RAM and the database in a SSD disk.

All the accuracy metrics indicated in this section are computed using the biometric model corresponding to **das-Face 3.2**. For more information, please read the *das-Face Performance Report.*

On identification, the system shows following accuracy and performance metrics:

- Accuracy is about 97.8% with MegaFace[6] dataset, using a gallery with 10,000 distractors, and using the EXACT accuracy configuration.
- The system is able to search over 1,000,000 identification candidates per second when using HIGH accuracy[7]. If using EXACT accuracy, the time needed is 5.8 seconds on average.

Following table shows the accuracy of the system depending on the number of elements in the gallery and the rank of the match. A rank-1 means that best match was found first in the list of candidates of the identification, and rank-10 means that best match was found up-to the first 10 candidates of the identification.

[6] http://megaface.cs.washington.edu/

[7] This speed is without considering the time to generate the embedding vector for the probe face image, and using HIGH accuracy on asynchronous identification. Using accuracy different than EXACT has a computational cost which is sublinear, so, when scaling down or up, computation time don't changes linearly.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Gallery size	*Rank*	
	1	*10*
10	98.7%	100.0%
100	98.2%	98.6%
1,000	98.0%	98.0%
10,000	97.8%	97.8%
100,000	94.3%	97.5%

Regarding clustering operations, the system shows following accuracy and performance metrics:

- The clustering accuracy is about 97.6% with setting=EXACT and a minConfidence=0.97 on LFW[8] benchmark dataset.[9] LFW dataset contains more than 13,000 images.
- Clustering performance is shown in the following table, depending on the gallery size and the indicated accuracy level.

	Accuracy Level			
Gallery size	*EXACT*	*HIGH*	*MEDIUM*	*LOW*
≅ 100	9 seconds	-	-	-
≅ 1,000	90 seconds	4 minutes	42 seconds	14 seconds
≅ 10,000	34 minutes	21 minutes	5 minutes	80 seconds
≅ 100,000	2 days	6 hours	1 hour	15 minutes

3.3. Data format considerations

The das-FaceBond system admits to upload data in form of independent images or packed in ZIP, TAR or TAR+GZIP packages (with mime types: application/x-tar, application/x-compressed-tar, application/zip, application/gzip).

[8] http://vis-www.cs.umass.edu/lfw/

[9] Consider this minConfidence just orientative, it is task dependent, and in some cases it will be required to use a higher or lower value, usually in range [0.95, 0.99].

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Images must be PNG or JPEG (with mime types: image/jpeg, image/png). It should be considered that images are JPEG and with a mean size of 100KB per image file in order to fulfill previously stated hardware requirements. PNG files are allowed, but they are not recommended because of their larger disk space requirements. Images are recommended to be around 1000 pixels in width.

Images should be sent in a straight position, that is, showing the face from top (head) to bottom (chin).[10] Faces should be 1/8th part of the total size of the image.[11]

When populating the galleries with image packages (ZIP or TAR), they must be of at most 500MB in size and not to include more than 20.000 images. Each image is recommended to have a filename which encodes a sort of identifier (e.g., customer identifier or similar). The system will register the following metadata fields per each image file found in the package:

- name: A field containing the whole path of the image in the package, including all directories.
- basename: Contains the filename of the image, removing all directory names from the path.
- package: The filename of the package where the image was uploaded into the system.

4. Container configuration

The container may be configured with a set of environmental variables which can be given when running it with docker or similar commands. The following is a list indicating default values for variables which have a default value.

4.1. Server and container behavior

- *SECRET_KEY*: It is a mandatory variable, which is used by the server for random numbers generation.
- *DEBUG*=**False**: It may be **True** or **False**, please, never use True on production environments.
- *ALLOWED_HOSTS*=*: This may be a list separated by commas of hosts from where requests are accepted. Each host may be given by an IP address or a domain name. No wildcards are allowed on host items. By default, it accepts requests from any host.
- *TZ*=**Europe/Madrid**: Configures the time-zone of the server.

[10] The system allows to rotate images in order to look for faces, but the activation of this feature means up to 4 times more computation when populating the galleries.
[11] This limitation can be relaxed by a parameter of the system. But changing this parameter from its default value requires more computational power when populating the galleries.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

- *WORKERS*=**4**: Number of threads to run in the docker container. The minimum number of workers must be **2**.
- ENABLE_SSL=**TRUE**: ssl enabled, exposing the service with HTTPS in additional port. By default it is set to False. It is important to note that the dasfacebond and the facebondui images come with self-signed certificates. They are just for illustrative purposes and are intended to be used for development and testing and never for production environments. They must be replaced with valid trusted certificates by mounting the certs folder as a volume as shown in the examples below, and saving there, the files server.crt and server.key.

4.2. Log configuration

- *ACTIVITY_ID_HTTP_HEADER*=**X-Request-Id**: Indicates a header which may be used to trace requests coming from another system. This header may be logged using JSON format.
- *LOG_LEVEL*=**INFO**: Default logging level, it can be **CRITICAL**, **ERROR**, **WARNING**, **INFO**, **DEBUG**.
- *LOG_FORMAT*=**console-simple**: Configures the way logging lines will be structured before written to the corresponding handler. It accepts the values **plain**, **console-simple**, **console**, **json**.
- *LOG_HANDLER*=**stdout**: Indicates where logging lines will be displayed. It can be **stdout** and **file**. When **stdout** is used, the following log variables will be ignored.
- *LOG_FOLDER*=**/tmp**: The folder where log file will be created.
- *LOG_FILENAME*=**dasfacebond.log**: The name of the log file where logging lines will be written.

4.3. Databases connections

- *DB_NAME*=**foo**: Name of the database where all tables and namespaces will be created.
- *DB_USER*=**foo**: Name of the user with granted privileges for creating tables and reading/writing data.
- *DB_PASS*=**foo**: Password for authentication of the user.
- *DB_HOST*=**localhost**: Host where the database server is located.
- *DB_SSLMODE*=**prefer**: Indicates how SSL connection to the database will be handled. It accepts **disable**, **allow**, **prefer**, **require**, **verify-ca**, **verify-full**.[12]
- *DB_SSLROOTCERT*: Location of the root certificate for SSL verification procedure.
- *LSH_DB_NAME*=$*DB_NAME*: Database name used for face embedding vector indices. By default is the same as *DB_NAME*. **Notice that *LSH_DB_USER* has to be granted with schema creation permission, because each gallery index requires a new schema in the database.**

[12] For more information, look at table 32-1 at https://www.postgresql.org/docs/9.6/static/libpq-ssl.html

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

- *LSH_DB_USER*=$*DB_USER*: Similar to *DB_USER* but for embedding vector indices.
- *LSH_DB_PASS*=$*DB_PASS*: Similar to *DB_PASS* but for embedding vector indices.
- *LSH_DB_HOST*=$*DB_HOST*: Similar to *DB_HOST* but for embedding vector indices.
- *LSH_DB_SSLMODE*=$*DB_SSLMODE*: Similar to *DB_SSLMODE*. Currently, this option exists but it is not fully implemented.
- *LSH_DB_SSLROOTCERT*=$*DB_SSLROOTCERT*: Similar to *DB_SSLROOTCERT*. Currently, this option exists but it is not fully implemented.

4.4. Email registration

- *SMTP_SERVER*: SMTP server for self-registration of new users using an email address.
- *SMTP_PORT*: Port for the connection with the SMTP server.
- *SMTP_FROM*: SMTP user name used for authentication and as sender of the email.
- *SMTP_PWD*: Password for authentication of *SMTP_FROM* user.

4.5. Redis connection

- *BROKER_HOST*=**localhost**: Host name of the broker used to handle Celery tasks. It should be the host of a Redis server.
- *BROKER_PORT*=**6379**: Port for connection with the Celery broker.
- *BROKER_SSL*=**False**: Indicates to use SSL for secure connections.
- *BROKER_SSLROOTCERT*: Path where SSL root certificate is located.
- *REDIS_HOST*=**localhost**: Host name of the Redis server used to cache service operations.
- *REDIS_PORT*=**6379**: Port for the connection with Redis server.
- *REDIS_DB*=**0**: Database to use to store service cache.
- *REDIS_SSL*=**False**: Indicates to use or not SSL for secured connections.
- *REDIS_SSLROOTCERT*: Path where SSL root certificate is located.

4.6. Identification parameters

- *TOP_MATCHES*=**10**: Number of match candidates to be persisted on the database for each identification.
- *IDENTIFICATION_ANNOTATIONS_LABOR_TIME*=**60**: Number of seconds a human agent requires to annotate an identification operation.

4.7. das-Face connection

- *FACE_BIOMETRICS_URL*=**http://localhost:5031**: Base URL to the server where das-Face API is available.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

- *FACE_BIOMETRICS_MODE*=**SelfieMode**: Default mode to communicate with das-Face. It can be **SelfieMode**, **DocumentMode**.
- *FACE_BIOMETRICS_LOCATE_MAX_SHAPE=**1000***: Any image with an edge greater than this value will be shrinked to this size on the larger edge. This parameter allows the user to control the performance when adding new faces to the server, but depending on this value, the system may fail to detect very small faces.
- *FACE_BIOMETRICS_LOCATE_ROTATIONS*=**False**: Indicates to rotate face image in case the system doesn't locate a face in it. Activating this option will make the system slower when no face is located in the image, but it will be tolerant to rotated images.

4.8. Proxy configuration

If you are required to deploy the server behind a proxy, and connections from the server to Veri-SaaS cloud are required, you need to add the following environment variables:

- *HTTP_PROXY*=http://127.0.0.1:3001
- *HTTPS_PROXY*=https://127.0.0.1:3001
- *NO_PROXY*=

Notice that such environment variables are necessary on-premises for das-Face product communication with Veri-SaaS cloud.

5. Images Installation

In order to install das-FaceBond system, it is required a machine with Ubuntu 18.04 LTS or RHEL 7.6 installed. In order to simplify the process, an Ansible playbook is given with the software delivery. Installation of Ansible and of all the required dependencies is performed by means of a shellscript (install-dependencies.sh).

This installation is run in two steps, installation of dependencies, and importation of docker images.

Depdendencies are installed by running the `install-dependencies.sh -o $OS -g $IS_GPU` script, where `$OS` can be ubuntu or rhel7.6 and $GPU can be yes or no, depending if the host used to deploy the product has a compatible GPU and its use is desired. This script will install Ansible with few additional dependencies, and executes an Ansible playbook which will install docker, nvidia-docker, docker-compose, nvidia-driver and all the required dependencies.

The second installation step is to import all docker images into the target machine, by running the `import-docker-images.sh`.

The following is an execution example of both scripts.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

$ ./install-dependencies.sh -o rhel7.6 -g yes

The whole system is delivered as a set of docker images:

- dasfacebond:1.6.1.tgz
- dasface:3.2.0-onpremises-gpu.tgz
- facebondui:1.0.2.tgz
- postgres:9.6.1.tgz
- redis:4.0.tgz

6. Deployment

The following is a description of how the deployment should be performed. A shellscript (run.sh) will be delivered, with the purpose of configuring and running the whole system. Configuration step blocks at the end, so it is required to press ctrl+C in order to continue the execution. In order to understand the process, this section explains all the steps involved in such a script.

This section is structured in two subsections, the first one explains the steps required to initialize the service the first time it is running (it migrates database and creates a new user and a new application), and a second section indicating how to deploy the system once everything has been previously configured.

6.1. Configuration

The following docker-compose script configures the docker container relying into another docker container for the database. You may ignore database container if you have one properly deployed on a server with SSD disks. The initial configuration is executed by a small script put into das-FaceBond docker image and in FaceBondUI docker image. This script has the following considerations:

- The script waits 10 seconds for startup of the postgresql database.
- */code/media* should be a volume where media files (images and TAR/ZIP packages) will be stored for persistence. The script requires this volume to change the ownership and group to be *www-data* (*uid*=33, *gid*=33). Doing so, once the service is started (next subsection), it will have permissions to write incoming images and packages.
    - Root privileges are necessary by the docker container to successfully execute *chown* command.
    - Such a media volume must be stored in a UNIX-like filesystem, in order to allow uid and gid to be replaced by the correct ones. The system won't work on SMB shared folders (or similar shared volumes where UNIX permissions and owner settings are not available).

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

- The script creates an admin user without login credentials, with name and email equal to "*admin@nodomain.com*".
- This script will perform the first database migration, which includes creation of tables and population of initial values on a few tables.
- After running this configuration process, the dasfacebond container should exit with 0 code. Any value different than 0 means that something bad happens during the configuration. Similarly, the facebondui container should exit with 0.
    - After both containers exited properly, it is required to press **ctrl+C** in order to put down the configuration docker composition.

```
version: '2.3'
services:

  dasfacebond:
                                                                    image:
registry.gitlab.com/veridas/face-team/products/dasfacebond:1.6.1
    entrypoint: /code/initializer.sh
    container_name: dasfacebond
   environment:
      CLIENT_ID: client_id
      CLIENT_SECRET: client_secret
      APP_NAME: app_name
      DEBUG=False
      SECRET_KEY=secret_key
      LOG_LEVEL=INFO
      TZ=Europe/Madrid
      DB_NAME=dasfacebond_database
      DB_USER=dasfacebond
      DB_PASS=dasfacebond
      DB_HOST=pgsql
      DB_SSLMODE=prefer
      WORKERS=6
      BROKER_HOST=redis
      BROKER_PORT=6379
      REDIS_HOST=redis
      REDIS_PORT=6379
      REDIS_DB=0
      FACE_BIOMETRICS_URL=http://dasface:8000
      FACE_BIOMETRICS_MODE=SelfieMode
      TOP_MATCHES=13
      DJANGO_SETTINGS_MODULE=app.settings
      ENABLE_SSL=TRUE
```


All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

```
  volumes:
    - ./media:/code/media
    - ./path_to_certs_folder:/etc/VERIDASsecurity/security/certs/

facebondui:
  image: registry.gitlab.com/veridas/face-team/products/facebondui:1.0.2
  entrypoint: /code/facebondui-initializer.sh
  volumes:
    - ./:/work:ro
  environment:
    CLIENT_ID: client_id
    CLIENT_SECRET: client_secret
    APP_NAME: app_name
    DEBUG=False
    SECRET_KEY=secret_key
    LOG_LEVEL=INFO
    TZ=Europe/Madrid
    OAUTH_CLIENT_ID=client_id
    OAUTH_CLIENT_SECRET=client_secret
    DB_NAME=dasfacebond_ui_database
    DB_USER=dasfacebond
    DB_PASS=dasfacebond
    DB_HOST=pgsql_ui
    DB_SSLMODE=prefer
    ENABLE_FACES=True
    SERVER_FACEBOND_API=http://dasfacebond:8820
    ENABLE_FACES=True
    ENABLE_PROBE_FACES=True
    WORKERS=2
    DJANGO_SETTINGS_MODULE=app.settings
    ENABLE_SSL=TRUE
volumes:
   - ./path_to_certs_folder:/etc/VERIDASsecurity/security/certs/

pgsql:
  image: postgres:9.6.1
  container_name: pgsql
  environment:
    TZ: Europe/Madrid
    POSTGRES_DB: dasfacebond_database
    POSTGRES_USER: dasfacebond
    POSTGRES_PASSWORD: dasfacebond
    PGDATA: /var/lib/postgresql/data/pgdata
```


All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

```
    volumes:
      - ./pgdata:/var/lib/postgresql/data/pgdata

  pgsql_ui:
    image: postgres:9.6.1
    container_name: pgsql_ui
    environment:
      TZ: Europe/Madrid
      POSTGRES_DB: dasfacebond_ui_database
      POSTGRES_USER: dasfacebond
      POSTGRES_PASSWORD: dasfacebond
      PGDATA: /var/lib/postgresql/data/pgdata
    volumes:
      - ./pgdata-ui:/var/lib/postgresql/data/pgdata
```

6.2. Service start

Once everything is configured, the service is ready to be started. The following docker-compose script shows how to run the service side-by-side with das-Face, Redis, Celery and the database containers. The das-Face container of this example is configured to use the nvidia driver version by using a docker runtime installed by nvidia-docker v2.

```
version: '2.3'
services:

  facebondui:
    image: registry.gitlab.com/veridas/face-team/products/facebondui:1.0.2
    container_name: facebondui
    environment:
      TZ=Europe/Madrid
      DEBUG=False
      SECRET_KEY=secret_key
      LOG_LEVEL=INFO
      OAUTH_CLIENT_ID=client_id
      OAUTH_CLIENT_SECRET=client_secret
      DB_NAME=dasfacebond_ui_database
      DB_USER=dasfacebond
      DB_PASS=dasfacebond
      DB_HOST=pgsql_ui
      DB_SSLMODE=prefer
      ENABLE_FACES=True
      SERVER_FACEBOND_API=http://dasfacebond:8820
      ENABLE_FACES=True
```


All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

```
    ENABLE_PROBE_FACES=True
    WORKERS=2
    DJANGO_SETTINGS_MODULE=app.settings
    ENABLE_SSL=TRUE
  ports:
    - 10000:8850
  restart: always
  runtime: nvidia
  volumes:
    - ./path_to_certs_folder:/etc/VERIDASsecurity/security/certs/

nginx_facebondui:
  image: registry.gitlab.com/veridas/face-team/products/facebondui:1.0.2
  container_name: nginx_facebondui
  environment:
    TZ=Europe/Madrid
    DEBUG=False
    SECRET_KEY=secret_key
    LOG_LEVEL=INFO
    OAUTH_CLIENT_ID=client_id
    OAUTH_CLIENT_SECRET=client_secret
    DB_NAME=dasfacebond_ui_database
    DB_USER=dasfacebond
    DB_PASS=dasfacebond
    DB_HOST=pgsql_ui
    DB_SSLMODE=prefer
    ENABLE_FACES=True
    SERVER_FACEBOND_API=http://dasfacebond:8820
    ENABLE_FACES=True
    ENABLE_PROBE_FACES=True
    WORKERS=2
    DJANGO_SETTINGS_MODULE=app.settings
    SERVER_NAME=server
    NGINX_UPSTREAM=facebondui
  command: /code/run_nginx.sh
  depends_on:
    - facebondui
  ports:
    - 8080:80
    - 8443:443
  restart: always
  runtime: nvidia
  volumes:
```


All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

```
      - ./path_to_certs_folder:/etc/VERIDASsecurity/security/certs

  dasfacebond:
                                                                    image:
registry.gitlab.com/veridas/face-team/products/dasfacebond:1.6.1
    container_name: dasfacebond
    environment:
      TZ=Europe/Madrid
      DEBUG=False
      SECRET_KEY=secret_key
      LOG_LEVEL=INFO
      DB_NAME=dasfacebond_database
      DB_USER=dasfacebond
      DB_PASS=dasfacebond
      DB_HOST=pgsql
      DB_SSLMODE=prefer
      WORKERS=6
      BROKER_HOST=redis
      BROKER_PORT=6379
      REDIS_HOST=redis
      REDIS_PORT=6379
      REDIS_DB=0
      FACE_BIOMETRICS_URL=http://dasface:8000
      FACE_BIOMETRICS_MODE=SelfieMode
      TOP_MATCHES=13
      DJANGO_SETTINGS_MODULE=app.settings
      ENABLE_SSL=TRUE
    ports:
      - 8820:8820
    volumes:
      - ./media:/code/media
      - ./path_to_certs_folder:/etc/VERIDASsecurity/security/certs/
    restart: always
    runtime: nvidia

  nginx_dasfacebond:
                                                                    image:
registry.gitlab.com/veridas/face-team/products/dasfacebond:1.6.1
    container_name: nginx_dasfacebond
    environment:
      TZ=Europe/Madrid
      DEBUG=False
      SECRET_KEY=secret_key
```


All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

```
      LOG_LEVEL=INFO
      DB_NAME=dasfacebond_database
      DB_USER=dasfacebond
      DB_PASS=dasfacebond
      DB_HOST=pgsql
      DB_SSLMODE=prefer
      WORKERS=6
      BROKER_HOST=redis
      BROKER_PORT=6379
      REDIS_HOST=redis
      REDIS_PORT=6379
      REDIS_DB=0
      FACE_BIOMETRICS_URL=http://dasface:8000
      FACE_BIOMETRICS_MODE=SelfieMode
      TOP_MATCHES=13
      DJANGO_SETTINGS_MODULE=app.settings
      SERVER_NAME=server
      NGINX_UPSTREAM=dasfacebond
    command: /code/run_nginx.sh
    depends_on:
      - dasfacebond
    ports:
      - 9999:80
      - 9443:443
    restart: always
    runtime: nvidia
    volumes:
      - ./path_to_certs_folder:/etc/VERIDASsecurity/security/certs

  dasface:
                                                                image:
registry.gitlab.com/veridas/face-team/products/dasface:3.2.0-onpremises-gp
u
    container_name: dasface
    environment:
      TZ=Europe/Madrid
      WORKERS=3
      PORT=8000
      DEBUG=no
      FLASK_DEBUG=0
      LOG_LEVEL=INFO
      DASFACES_DEFAULT_GPU_MEMORY_FRACTION=0.2
      LD_LIBRARY_PATH=/usr/local/nvidia/lib64:/usr/local/cuda/lib64
```


All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

```
      USAGE_TRACKER_DEPLOY_ENV=production
      USAGE_TRACKER_API_KEY=APIKEY
      OMP_NUM_THREADS=1
    ports:
      - 2000:8000
    restart: always
    runtime: nvidia

  redis:
    image: redis:4.0
    container_name: redis
    command: redis-server
    environment:
      TZ=Europe/Madrid
    restart: always
    runtime: nvidia

  celery:
                                                                    image:
registry.gitlab.com/veridas/face-team/products/dasfacebond:1.6.1
    command: ./run-celery-worker.sh
    container_name: celery
    user: "www-data"
    environment:
      TZ=Europe/Madrid
      DEBUG=False
      SECRET_KEY=secret_key
      LOG_LEVEL=INFO
      DB_NAME=dasfacebond_database
      DB_USER=dasfacebond
      DB_PASS=dasfacebond
      DB_HOST=pgsql
      DB_SSLMODE=prefer
      WORKERS=6
      BROKER_HOST=redis
      BROKER_PORT=6379
      REDIS_HOST=redis
      REDIS_PORT=6379
      REDIS_DB=0
      FACE_BIOMETRICS_URL=http://dasface:8000
      FACE_BIOMETRICS_MODE=SelfieMode
      TOP_MATCHES=13
      DJANGO_SETTINGS_MODULE=app.settings
```


All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

```
    volumes:
      - ./media:/code/media
    depends_on:
      - redis
    restart: always
    runtime: nvidia

  pgsql:
    image: postgres:9.6.1
    container_name: pgsql
    environment:
      TZ: Europe/Madrid
      POSTGRES_DB: dasfacebond_database
      POSTGRES_USER: dasfacebond
      POSTGRES_PASSWORD: dasfacebond
      PGDATA: /var/lib/postgresql/data/pgdata
    volumes:
      - ./pgdata:/var/lib/postgresql/data/pgdata

  pgsql_ui:
    image: postgres:9.6.1
    container_name: pgsql_ui
    environment:
      TZ: Europe/Madrid
      POSTGRES_DB: dasfacebond_ui_database
      POSTGRES_USER: dasfacebond
      POSTGRES_PASSWORD: dasfacebond
      PGDATA: /var/lib/postgresql/data/pgdata
    volumes:
      - ./pgdata-ui:/var/lib/postgresql/data/pgdata
```

6.3. Users creation

Once the service is up and running, you may want to create new users, staff and/or superusers (admins). You may use the *manage.py* command to do that in the *dasfacebond* container. Something like the following will create new superusers:

```
$ docker exec -t -i dasfacebond python manage.py createsuperuser
```

You may ask the help of this command by running:

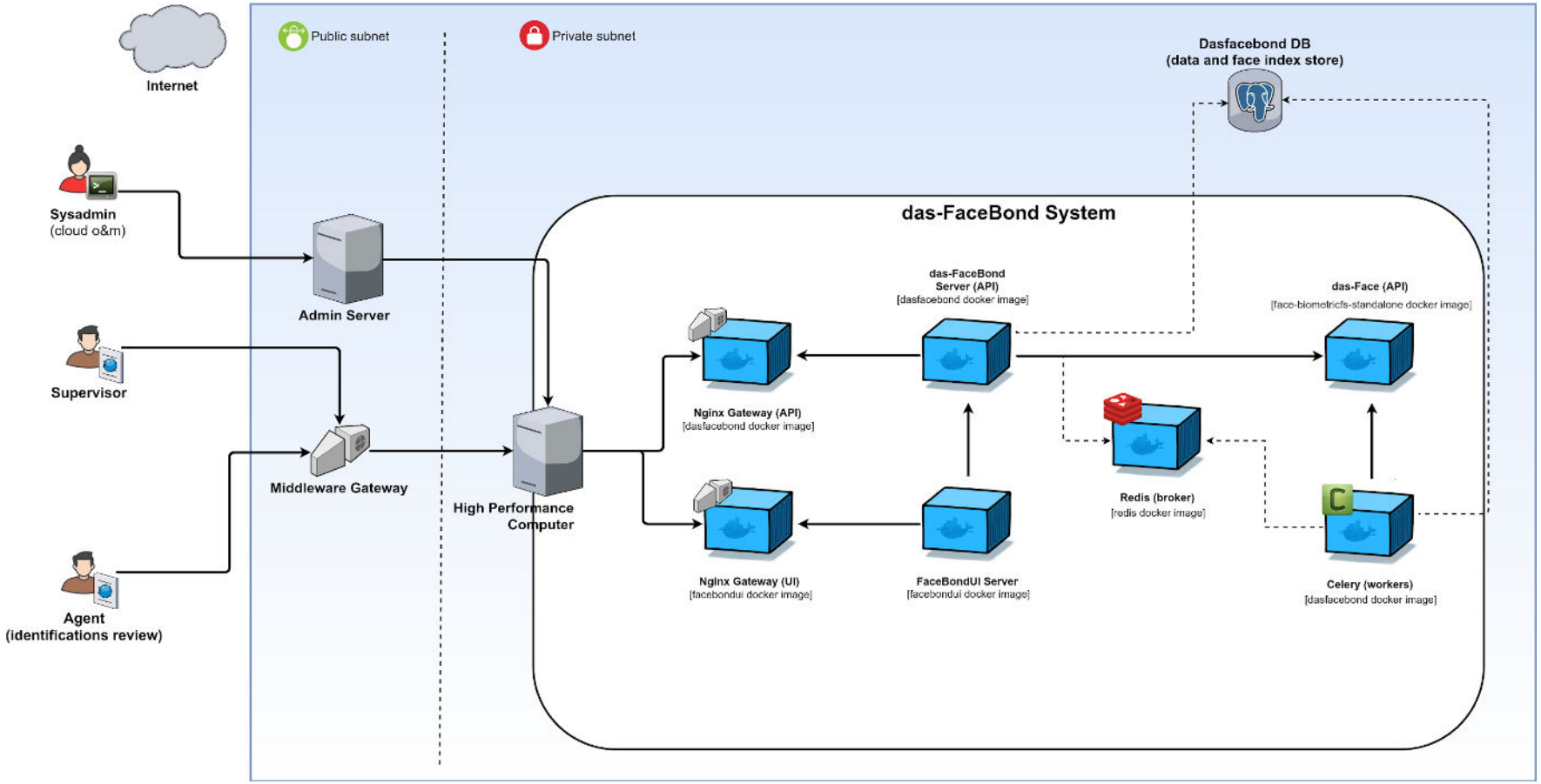
```
$ docker exec -t -i dasfacebond python manage.py createsuperuser --help
```


All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Once you have an account, it is possible to login into the system to create accounts for new users (admin or staff) and to register users in group **agent**, used for exploitation of manual review procedures incorporated in das-FaceBond, as explained in the appendix.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

6.4. Containers Deployment Diagram



All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

A. Appendix - General Use Cases

A.0. User Roles

The system allows to create users in two different roles:

- admin: users with this role belong to user group ‘admin’, and they are able to execute any of the endpoints of this API.
    - Supervisors (people who will review the job of agents) will play the same role as admins.
- agent: users with this role belong to user group ‘agent’, and they are limited to endpoints related with identification review process.

New users may be created using the admin interface deployed in the nginx gateway, under the path **/admin**. It is possible to create new superusers by running the manage.py script or by using the nginx gateway to admin interface. Superusers have both roles, ‘admin’ and ‘agent’.

A.1. Image Upload

Images should be uploaded before adding a face to a gallery, and before running an identification operation. When adding a new face, the uploaded image UUID should be passed to the face creation endpoint. When running a new identification, the uploaded image UUID should be passed to the identification endpoint.

A.2. Package Upload

A package with a bunch of images may be uploaded for population of galleries. This way, the client could insert a large number of faces in the gallery, just by uploading the package and using the returned UUID as input of the face batch operation endpoint.

A.3. Create a new Gallery

Galleries may be created and populated on demand. Any gallery contains a set of faces, no consideration about the identity of the faces exists in the system, beyond a custom metadata associated with each face. Galleries are totally disjoint, face sharing is not allowed. For gallery creation, just a name, a description and an operation mode are required. During gallery creation a query to das-Face service will be performed, selecting the last biometric model for future faces addition.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

A.4. Populate a Gallery

A gallery may be populated following two procedures:

- Adding faces one by one. An image upload will be required, and then, a new face association may be done.
- Adding faces on batch. A package upload will be required, and then, all face images contained in the package will be inserted as images and associated with the gallery.

During any of both processes, das-Face will be used to generate face embeddings using the model annotated when the gallery was created.

A.5. An Identification

An identification requires two components: a probe image (containing a face); and a target gallery. The probe image should be uploaded first, and then the operation will be requested. The operation comprises the comparison of the probe versus all matching candidates in the target gallery. The list of matching candidates may be filtered by a minimum match threshold, and it may be filtered based on locality algorithms for performance reasons.

A.6. Batch of Identifications

A batch of identifications requires two components: a gallery with probe faces; and a target gallery. All images, taken from the faces of the probe gallery, will be compared to all the faces in the target gallery. Each one of the identifications will follow the same procedure as described at A.5.

A.7. Clustering of a Gallery

Gallery faces may be clustered together in natural groups based on similarity between them. This operation is useful to look for similar faces, or a person with more than one face in the gallery. This operation requires a target gallery, a threshold for minimum similarity, and an expected accuracy indication. Depending on the accuracy, the operation may be faster, and less accurate, or slower but more accurate.

A.8. Review of identifications

The das-FaceBond system comes with an implementation which facilitates the manual review[13] of identification operations by a human agent. This review process annotates identifications with three label types:

[13] As much large is the gallery of faces, much important is the manual review of the process.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

- Review decision state: It is indicated by the human agent and it is stored in the database. This decision can be:
    - NOT_OPERATIONAL: when the identification operation is running, or it has failed for some reason.
    - PENDING: when the identification operation has finished and it is ready to be reviewed by a human agent.
    - DONE_CONCLUSIVE: when the human agent has annotated the operation and he was sure about his decisions.
    - DONE_DOUBTFUL: when the human agent has annotated the operation but he has doubts about his decisions.
- Candidates: It is a list of faces marked as matches with the probe image used in the identification.
- Observations: A free text indicating any kind of commentary given by the human agent.

Pending identification operations will be served in FIFO order to be reviewed by human agents. To avoid multiple agents reviewing the same operation, the time required to review the identification is indicated at variable IDENTIFICATION_ANNOTATIONS_LABOR_TIME, in seconds, and this time a served operation will be kept frozen in the queue.

The Figure 1 shows an automaton of transitions between review decision states. The system will validate the indicated annotations to be consistent with the automaton valid transitions.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

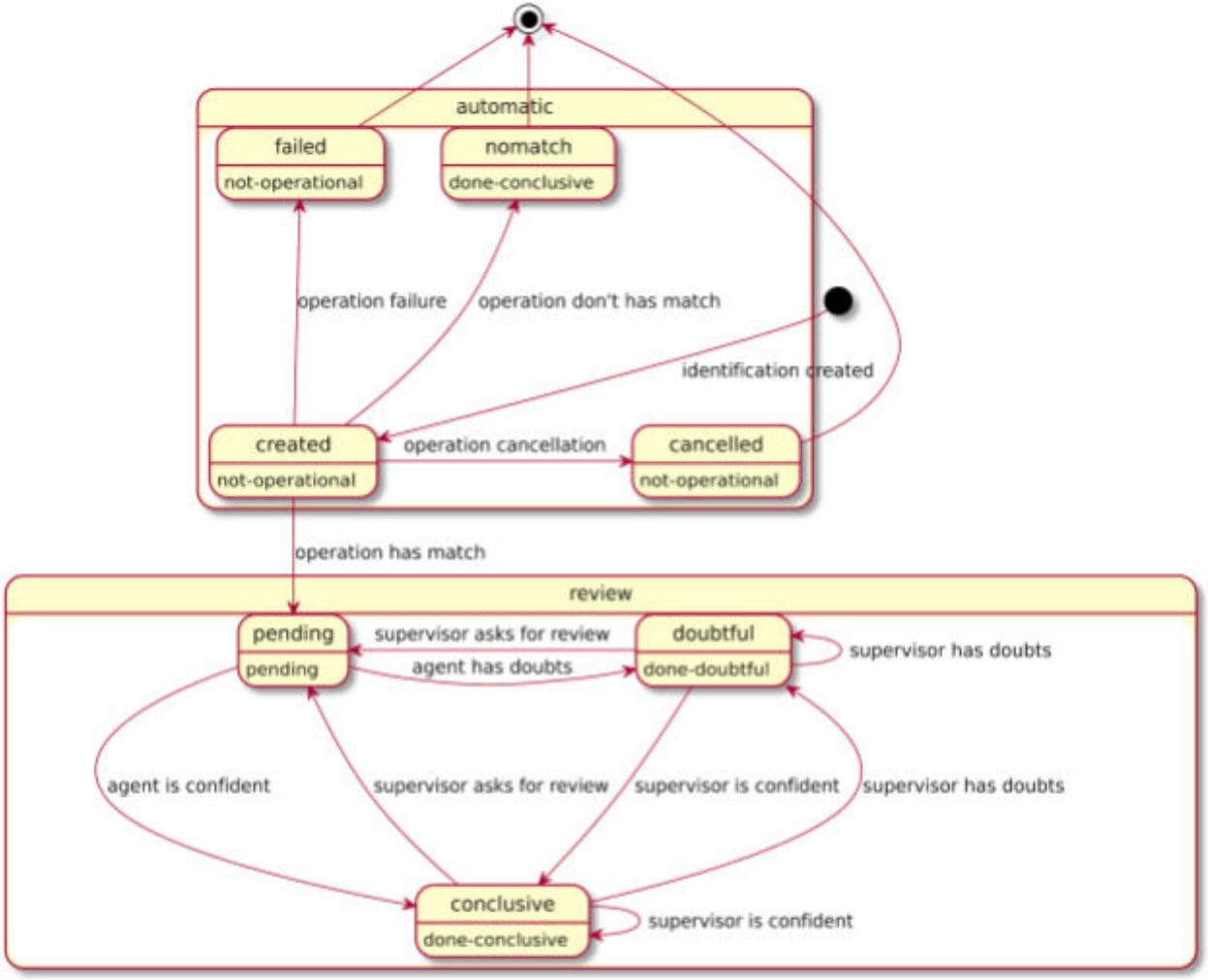


Figure 1. Automaton representing transitions of decision review state of identification annotations. Each automaton state contains the operation status (top) and the decision state (bottom). Transitions indicates the executor user and the executed action.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

B. Appendix - API

Requests to the server can be in application/json, and some endpoints accept as well multipart/form-data. Responses are always JSON, and in case of a failure, the server response is a JSON with the following fields:

Field	Required	Description
exception	no	Error code, for example: `ValidationError`, `FaceNotFoundError`, etc.
message	no	A message providing more information about what went wrong. The message may be free and not specified for certain error types.
FIELDNAME	no	Missing or mistaken fields on a POST request. Each field will be an item in the dictionary.

Example:

```
{
      "exception": "FaceNotFound",
      "message": "Unable to locate a face in the image"
}
```

Validation errors (requests with missing or badly-formed fields) will be returned with a 400 HTTP status code and as a dictionary where field names are keys and values are arrays of strings with messages related to errors on the corresponding field. These messages won't contain 'exception' or 'message' fields in the response. In the following documentation, these messages will be indicated as ValidationError in the following.

Some of the objects returned by this API can be paginated, i.e., they are queried and returned in paginated format. This format indicates the current page (page), the number of objects per page (perPage), the total number of pages (numPages) and the total number of objects (count). Additionally, this format also indicates the type of objects that are paginated. The paginated array will be formed by items of a particular resource, and the name of the array is the type of resource in plural. Then, paginated responses are JSON based with the following fields:

Field	Required	Description
page	yes	Current page.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

perPage	yes	Number of items per page.
numPages	yes	Number of pages in total.
count	yes	The total number of items, so numPages = count / perPage
ARRAYNAME	yes	Name of the array, for instance, in the case of galleries it is ARRAYNAME=galleries

This paginated responses will be documented by indicating the ARRAYNAME value and the content of just one of the array items.

The following example shows page 1 of paginated Gallery objects where each page may contain up to 10 galleries. The total amount of galleries being paginated is 5 so the total number of pages is 1.

Example:

```
{
    "page": 1,
    "perPage": 10,
    "numPages": 1,
    "count": 5,
    "galleries": [
        {
            "galleryId": "58da7427-cf42-487a-ab6b-0d2717c3c492",
            …,
        },
        {
            "galleryId": "20a6e6f7-a3d9-4ffb-959c-72b45f1b01f4",
            …,
        }
    ]
}
```

B.1. API v1

All API v1 endpoints are prefixed with **/api/v1.** This section is structured by resources available in the API v1:

- Images: this resource allows CRUD operations on images, which once uploaded into the system, may be used for identification or gallery face manipulation.
- Packages: this resource allows CRUD operations on packages with images (TAR or ZIP packages). They may be used to populate galleries with a large batch of faces.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

- Galleries: this resource allows CRUD operations on groups (galleries) and faces inside these galleries.
    - Faces: each face is assigned to a gallery, and a face contains metadata describing customer-oriented information.
    - FaceBatches: faces may be uploaded in batches, where each batch populates a gallery with the images contained in a Package object.
- Identifications: it contains all comparisons of a probe image to a target gallery of faces (previously populated). This resource may run operations in a synchronous or asynchronous way. Synchronous operations are exact, but limited to a number of faces in the gallery. Asynchronous operations may work with larger databases, and they allow configuring the accuracy level.
    - Matches: identification operation produces match candidates, sorted by similarity between the probe and the target.
    - Annotations: labels given to the identification operation, allowing to subjectively identify matches in the list of candidates returned by the system.
- Clusterings: inference of natural groups on a gallery of faces. All clustering operations are performed asynchronously because they are intense in computational terms.
    - Clusters: each cluster is assigned to a clustering operation, and they contain a relation of faces put together by the clustering algorithm.
        - ClusterFace: clusters are composed of faces, which are basically a specialization of a Face object, but adding new fields to consider the face-in-cluster and the cluster-in-clustering relations.

B.1.1. Resources Declaration

This section introduces the structure (fields or properties) of the resources handled by das-FaceBond service. The API exposes methods to create, remove, update and delete these resources from the system. In the API documentation there may be references to these object structure declarations.

- Image objects contain a reference to the image in the server.

Returned data	**Required**	**Type**	**Description**
imageId	yes	UUID string	The UUID for identification of this object.
imageUrl	yes	string	The URL where the image may be retrieved from the server.
createdAt	yes	datetime string	A datetime string in ISO 8601 format.

- Package object keeps track of the package name used to upload the data.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Returned data	Required	Type	Description
packageId	yes	UUID string	The UUID for identification of this object.
originalFilename	yes	string	String containing the name of the package file as given on creation.
createdAt	yes	datetime string	A datetime string in ISO 8601 format.

- Gallery object has some descriptive fields and a mode of operation in das-Face. Additionally, it contains a reference to this gallery faces resource in the server.

Returned data	Required	Type	Description
galleryId	yes	UUID string	The UUID for identification of this object.
name	yes	string	The name given to this gallery. Be careful, only names with alphanumeric characters are allowed. The name is unique over all objects.
description	yes	string	A plain description given when the resource was created.
createdAt	yes	datetime string	A datetime string in ISO 8601 format.
mode	yes	string	Operation mode in das-Face: SelfieMode or DocumentMode
facesUrl	yes	URI string	The URI where all faces of this resource are located (list of faces).
numFaces	yes	integer	The number of counted faces in the gallery.

- Face object keeps track of custom face metadata and the image associated with the face.

Returned data	Required	Type	Description
faceId	yes	UUID string	The UUID for identification of this object.
galleryId	yes	UUID string	UUID of the gallery where this face belongs.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

image	yes	Image object	The Image object used to create this Face.
createdAt	yes	datetime string	A datetime string in ISO 8601 format.
metadata	yes	JSON object	A JSON object containing customer data associated with this Face. The metadata property cannot be recursive, it only may contain strings and/or numbers.

- FaceBatch object connects a package of data with a gallery, and indicates the status of the batch operation.

Returned data	**Required**	**Type**	**Description**
faceBatchId	yes	UUID string	The UUID for identification of this object.
galleryId	yes	UUID string	UUID of the gallery where this face belongs.
status	yes	string	One of: PENDING, FINISHED, FAILED, CANCELED.
completed	yes	integer	Number of faces inserted into the gallery. This value is updated while the batch is in PENDING state.
failed	yes	integer	Number of images failed to locate a face in it.
length	no	integer	Number of images located in the package.
packageId	yes	UUID	The UUID of the package used for this batch insertion.
packageFilename	yes	string	Filename of the package.
createdAt	yes	datetime string	A datetime string in ISO 8601 format.
startedAt	yes	datetime string	A datetime string in ISO 8601 format.
finishedAt	yes	datetime string	A datetime string in ISO 8601 format.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

- Identification objects contain data related to the comparison of a probe image with a gallery of faces.

Returned data	Required	Type	Description
identificationId	yes	UUID string	The UUID for identification of this object.
image	yes	Image object	The Image object used as probe of the identification.
gallery	yes	Gallery object	The Gallery object used as target of the identification.
minConfidence	yes	number	A threshold used to constraint the identification operation.
accuracyLevel	no	string	One of: EXACT, HIGH, MEDIUM, LOW. This string indicates the accuracy of the search, more accurate search requires more computation time.
status	yes	string	One of: MATCH, NO_MATCH, FAILED, PENDING. This string indicates if an identification finished successfully (MATCH), if it fail to match with the gallery (NO_MATCH), if it and if the process is enqueued or running (PENDING).
reason	no	string	A message
createdAt	yes	datetime string	A datetime string in ISO 8601 format.
finishedAt	no	datetime string	A datetime string in ISO 8601 format.
length	no	integer	Number of effective faces to compare.
rank1	no	Match object	The match with maximum similarity.
topMatchesCount	no	integer	Number of top matches. This count doesn't includes the rank1 match.
countGtConfidence	no	integer	Number of faces of the gallery found above the given minConfidence threshold.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

topMatches	no	array of Match objects	An array with topMatchesCount items.
metadata	no	JSON object	A JSON object containing customer data associated with this Face. The metadata property cannot be recursive, it only may contain strings and/or numbers.
annotation	yes	Annotation object	An instance of Annotation object.

- Match object indicates the confidence of the comparison between a probe (in an identification operation) and a face in the gallery.

Returned data	**Required**	**Type**	**Description**
confidence	yes	number	The similarity between the identification probe and the indicated face.
order	yes	integer	Order of this face in the list of matches above the minConfidence threshold.
face	yes	Face object	The Face object corresponding to current match.

- Annotation object indicates labels given by a human agent, and indicates the review state of the identification operation. The operation may be in these states:
    - NOT_OPERATIONAL: when the identification cannot be reviewed by a human agent, perhaps due to it being running, or perhaps the operation failed somehow.
    - PENDING: when nobody has reviewed the operation.
    - DONE_CONCLUSIVE: when the operation has been reviewed and the agent said he was sure about his decision.
    - DONE_DOUBTFUL: when the operation has been reviewed and the agent said he was doubtful about his decision.

Returned data	**Required**	**Type**	**Description**
annotationId	yes	UUID string	The UUID for this object.
candidates	no	Array of UUID strings	An array of UUID strings with faceId of candidates marked by the human agent as match with the probe image.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

observations	no	Text	A text written by the human agent to clarify anything about his decision process.
user	yes	string	Name the user responsible of the current Annotation object. When no human agent has reviewed the system, the user will be the one who run the identification operation.

- Clustering object keeps track of clustering operation configuration.

Returned data	**Required**	**Type**	**Description**
clusteringId	yes	UUID string	The UUID for identification of this object.
galleryId	yes	UUID string	UUID of the gallery where this face belongs.
gallery	yes	Gallery object	The Gallery object used as target for the clustering operation.
accuracyLevel	yes	string	One of: EXACT, HIGH, MEDIUM, LOW. Indicates how accurate is the clustering result. More accuracy translates into more time for the computation.
minConfidence	yes	number	Threshold to filter-out pair of faces below this similarity.
createdAt	yes	datetime string	A datetime string in ISO 8601 format.
startedAt	no	datetime string	A datetime string in ISO 8601 format.
finishedAt	no	datetime string	A datetime string in ISO 8601 format.
numClusters	no	integer	Number of clusters found by the algorithm.
status	yes	string	One of: PENDING, RUNNING, FINISHED, FAILED.
clustersUrl	yes	string	A URL string with the location of clusters list for current clustering operation.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

failureCode	no	string	In case of status=FAILED, this indicates a code with the kind of failure.

- Cluster object contains information related to a particular cluster in a clustering operation.

Returned data	**Required**	**Type**	**Description**
clusterId	yes	UUID string	The UUID for identification of this object.
clusteringId	yes	UUID string	UUID of the clustering operation which found this cluster.
galleryId	yes	UUID string	UUID of the gallery where this face belongs.
numFaces	yes	integer	Number of faces in the cluster.
faces	yes	array of Face objects	An array of face objects.

- ClusterFace object is a specialization of Face object, adding information related to the clustering operation and the cluster where the face belongs.

Returned data	**Required**	**Type**	**Description**
faceId	yes	UUID string	The UUID for identification of this object.
galleryId	yes	UUID string	UUID of the gallery where this face belongs.
clusterId	yes	UUID string	The UUID of the cluster where this face belongs.
clusteringId	yes	UUID string	UUID of the clustering operation which found this cluster.
image	yes	Image object	The Image object used to create this Face.
createdAt	yes	datetime string	A datetime string in ISO 8601 format.
metadata	yes	JSON object	A JSON object containing customer data associated with this Face. The metadata property cannot be recursive, it only may contain strings and/or numbers.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

B.1.1. Service Life Check and Authentication Methods

This service exposes an API with the following features:

- **Is alive**: The service receives a GET request with no params, and returns a 204 status code indicating that the server is up.

`GET` `/api/v1/alive`

Errors:

Code	HTTP Status	Message
`UnexpectedError`	500	An unknown error has occurred while processing the request

- **User registration**: Registers a user which will be allowed to work with the API. This endpoint only works when the SMTP server is properly configured.

`POST` `/api/v1/register`

Request:

Content-Type: application/json, application/x-www-form-urlencoded

Name	Req.	Type	Description
email	yes	string	Email of the user that is to be registered

Response: 200 status

Returns success response via status 200.

```
{
        “status”: “success”
}
```

Errors:

Code	HTTP Status	Message
ValidationError	400	-
PermissionDeniedError	403	-
RegisterError	409	Email already registered
UnexpectedError	500	An unknown error has occurred while processing the request

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

- **Authentication**: Authenticates a user returning their access token data. It is possible to authenticate superusers created by means of the docker container commands, and users self-registered by means of /register endpoint.

`POST` `/api/v1/auth`

Request:

Content-Type: application/json, application/x-www-form-urlencoded

Name	Req.	Type	Description
email	yes	string	Email of the registered user. API users must have an equal name and email.
code	yes	string	Code sent to user's email upon registration, or password of the user in the database
clientId	yes	string	Client ID of the OAuth2 application, as given to the initialization container
clientSecret	yes	string	Client secret of the OAuth2 application, as given to the initialization container
scope	no	string	A string indicating the scopes you desire to the received accessToken. Scopes are directly related with user roles. It must be a comma separated list of roles, for instance "admin,agent". If not given, it will be a list of all available user roles.

Response: application/json

Returned data	Type	Description
accessToken	string	Token to be used in authorization header
tokenType	string	Type of the token, usually contains "Bearer" string
expiresIn	int	How many seconds for expiration
refreshToken	string	This field is returned but currently not used by this API
scope	string	A string indicating the scopes active to the received accessToken. Scopes are directly related with user roles. It may be a comma separated list of roles, for instance "admin,agent", when more than role is available for the given token.

Errors:

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Code	HTTP Status	Message
ValidationError	400	-
PermissionDeniedError	403	-
UnexpectedError	500	An unknown error has occurred while processing the request

B.1.2. Image Methods

`GET /api/v1/images`

The service receives a GET request and returns a paginated list of available Images. It is possible to filter the list by using the indicated query parameters.

Headers	Content
Authorization	Bearer ACCESS_TOKEN

Query parameters:

Name	Description
page	Indicates which page of the array will be retrieved.
perPage	Number of items retrieved per page.
createdAtFrom	A date and time in ISO 8601 format, for filtering only items with a creation date posterior to this value.
createdAtTo	A date and time in ISO 8601 format, for filtering only items with a creation date inferior to this value.

Response: application/json

This response is paginated with ARRAYNAME=images, and where each item is an Image object.

Errors:

Code	HTTP Status	Message
PermissionDeniedError	403	-
UnexpectedError	500	An unknown error has occurred while processing the request

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

`POST` `/api/v1/images`
Creates a new Image resource.

Request:
Content-Type: multipart/form-data, media/jpeg, media/png

Headers	Content
Authorization	Bearer ACCESS_TOKEN

Body:

- When the content type is media/jpeg or media/png, it is expected a binary body containing the whole image data. In this case, it is required to add the header Content-Diposition header indicating the name of the image file:
  - Content-Disposition: attachment; filename=image.jpg
- For multipart/form-data requests, it is necessary to put following fields:

Name	Req.	Type	Description
file	yes	file	A binary file embedded as a part of the multi-part form.

Response: application/json (HTTP status 201)
This response returns the Image object as created in the system.

Errors:

Code	HTTP Status	Message
PermissionDeniedError	403	-
ValidationError	400	-
UploadError	415	Unsupported media type.
UnexpectedError	500	An unknown error has occurred while processing the request

`GET` `/api/v1/images/{imageId}`
Returns the given Image content, may be normalizing its aspect ratio and/or size as indicated in the query parameters.

Headers	Content

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Authorization	Bearer ACCESS_TOKEN

Query parameters:

Name	Description
ar	Aspect ratio of the returned image. The original image will be processed to ensure the returned image has the indicated aspect ratio, but without distorting the contained face. The ratio is given as '3:4' or '9:16' strings. By default it is '3:4'.
width	Indicates the width of the returned image. It will be transformed to have the indicated width, and height corresponding to the indicated aspect ratio parameter. By default it is empty, so the width of the original image will be used.
raw	When given, both previous parameters will be ignored, and the image as it is stored in the server will be returned.

Response: media/jpeg, media/png
The binary content of the image. This URL may be used to link images stored in the server.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownImageError`	404	Unable to locate given image UUID
`InvalidQueryParameter`	409	A not valid content for the query parameter has been given.
UnexpectedError	500	An unknown error has occurred while processing the request

`DELETE /api/v1/images/{imageId}`
Deletes the indicated image from the server.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	Content

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

`Authorization`	Bearer ACCESS_TOKEN

Response: Empty response (HTTP status 204)

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	
`UnknownImageError`	404	Unable to locate given image UUID
UnexpectedError	500	An unknown error has occurred while processing the request

B.1.3. Package Methods

`GET` `/api/v1/packages`

The service receives a GET request and returns a paginated list of available Pacakges. It is possible to filter the list by using the indicated query parameters.

Headers	Content
Authorization	Bearer ACCESS_TOKEN

Query parameters:

Name	Description
page	Indicates which page of the array will be retrieved.
perPage	Number of items retrieved per page.
createdAtFrom	A date and time in ISO 8601 format, for filtering only items with a creation date posterior to this value.
createdAtTo	A date and time in ISO 8601 format, for filtering only items with a creation date inferior to this value.

Response: application/json

This response is paginated with ARRAYNAME=batchPackages, and where each item is a Package object.

Errors:

Code	HTTP Status	Message

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

PermissionDeniedError	403	-
UnexpectedError	500	An unknown error has occurred while processing the request

`POST` `/api/v1/packages`
Creates a new Package resource.

Request:
Content-Type: multipart/form-data, application/x-tar, application/x-compressed-tar, application/zip, application/gzip

Headers	**Content**
Authorization	Bearer ACCESS_TOKEN

Body:

- When the content type is application/*, it is expected to be a binary body containing the whole package file data. In this case, it is required to add the header Content-Diposition header indicating the name of the package file:
  - Content-Disposition: attachment; filename=package.zip
- For multipart/form-data requests, it is necessary to put the following fields:

Name	**Req.**	**Type**	**Description**
file	yes	file	A binary file embedded as a part of the multi-part form.

Response: application/json (HTTP status 201)
This response returns the Package object as created in the system.

Errors:

Code	**HTTP Status**	**Message**
PermissionDeniedError	403	-
ValidationError	400	-
UploadError	415	Unsupported media type.
UnexpectedError	500	An unknown error has occurred while processing the request

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

B.1.4. Gallery Methods

- **Galleries**: Allows to create and list galleries on the system. A gallery is an object which allows to put together a bunch of face images. Once a gallery is ready, the client can perform identifications against all the faces in the gallery, and/or perform clustering operations for analysis and knowledge extraction from the data.

`GET /api/v1/galleries`

The service receives a GET request and returns a paginated list of available Galleries. It is possible to filter the list by using the indicated query parameters. For convenience, this list incorporates a link to the resource with all faces belonging to each gallery.

Headers	**Content**
Authorization	Bearer ACCESS_TOKEN

Query parameters:

Name	**Description**
page	Indicates which page of the array will be retrieved.
perPage	Number of items retrieved per page.
search	A search string to constraint retrieved data. This string will look for matches in gallery names.
createdAtFrom	A date and time in ISO 8601 format, for filtering only items with a creation date posterior to this value.
createdAtTo	A date and time in ISO 8601 format, for filtering only items with a creation date inferior to this value.

Response: application/json

This response is paginated with ARRAYNAME=galleries, and where each item is a Gallery object.

Errors:

Code	**HTTP Status**	**Message**
PermissionDeniedError	403	-
UnexpectedError	500	An unknown error has occurred while processing the request

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

`POST` `/api/v1/galleries`
Creates a new Gallery resource.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	Content
Authorization	Bearer ACCESS_TOKEN

Body:

Name	Req.	Type	Description
name	yes	string	Name given to the gallery. It is constrained to alphanumeric characters. It should be unique over all gallery objects.
description	yes	string	A plain description for this gallery.
mode	yes	DasFaceMode	A string containing SelfieMode or DocumentMode

Response: application/json (HTTP status 201)
This response returns the Gallery object as created in the system.

Errors:

Code	HTTP Status	Message
PermissionDeniedError	403	-
ValidationError	400	-
UnexpectedError	500	An unknown error has occurred while processing the request

`GET` `/api/v1/galleries/{galleryId}`
Returns data for the given Gallery.

Headers	Content
Authorization	Bearer ACCESS_TOKEN

Response: application/json
A Gallery object will be returned.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery is not available
UnexpectedError	500	An unknown error has occurred while processing the request

PATCH `/api/v1/galleries/{galleryID}`
This endpoint allows updating fields of the indicated Gallery object.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	Content
Authorization	Bearer ACCESS_TOKEN

Body:

Name	Req.	Type	Description
name	no	string	Replaces gallery name by this one.
description	no	string	Replaces gallery description by this one.

Response: application/json
A Gallery object will be returned.

Errors:

Code	HTTP Status	Message
ValidationError	400	-
PermissionDeniedError	403	-
`UnknownGalleryError`	404	Given gallery is not available
UnexpectedError	500	An unknown error has occurred while processing the request

PUT `/api/v1/galleries/{galleryID}`

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Replaces writable Gallery data.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Body

Name	Req.	Type	Description
name	yes	string	New name for the gallery.
description	yes	string	New description for the gallery.

Response: application/json
A Gallery object will be returned.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery is not available
UnexpectedError	500	An unknown error has occurred while processing the request

`DELETE` `/api/v1/galleries/{galleryID}`
Deletes the given Gallery.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Response: Empty response (HTTP status 204)

Errors:

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery is not available
UnexpectedError	500	An unknown error has occurred while processing the request

B.1.4. Face Methods

- **Faces**: Lists and creates faces on a particular gallery .The gallery is given as part of the URI. All faces created there will belong to the particular gallery. Metadata is used on faces to store customer information. The expected schema is a flat object with all key and values with string type.

`GET /api/v1/galleries/{galleryID}/faces`
Returns all faces of the specified Gallery specified.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Query parameters:

Name	Description
page	Indicates which page of the array will be retrieved.
perPage	Number of items retrieved per page.
search	A search string to constraint retrieved data. This string will look for matches in metadata JSON content.
createdAtFrom	A date and time in ISO 8601 format, for filtering only items with a creation date posterior to this value.
createdAtTo	A date and time in ISO 8601 format, for filtering only items with a creation date inferior to this value.

Response: application/json, application/x-www-form-urlencoded
This method returns a paginated list of Face objects where ARRAYNAME=faces.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery is not available
UnexpectedError	500	An unknown error has occurred while processing the request

POST `/api/v1/galleries/{galleryID}/faces`
Inserts a new face at the indicated Gallery.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Body:

Name	Req.	Type	Description
metadata	yes	JSON object	A plain JSON object, only containing numbers and strings. This data content is free, so the client may decide what to persist here.
imageId	yes	UUID	The UUID of a previously uploaded Image object.

Response: application/json
This method returns the Face object as created on the system.

Errors:

Code	HTTP Status	Message
ValidationError	400	-
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery is not available
`FaceNotFoundError`	415	Unable to locate a face in the image.
`FaceAlignmentError`	415	The face cannot be normalized and aligned.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

`MoreThanOneFaceError`	415	More than one relevant faces are present in the image.
UnexpectedError	500	An unknown error has occurred while processing the request

GET `/api/v1/galleries/{galleryId}/faces/{faceId}`
Returns all data of resource Face identified by faceId and belonging to the specified gallery.

Request:

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Response: application/json
The content of the indicated Face object.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery is not available
`UnknownFaceError`	404	Given face is not available
UnexpectedError	500	An unknown error has occurred while processing the request

PATCH `/api/v1/galleries/{galleryId}/faces/{faceId}`
Updates fields of the indicated Face object.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Name	Req.	Type	Description
metadata	no	JSON object	A new metadata content to replace the one existing in the face object. The metadata will be replaced as a whole.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

imageId	no	UUID	The UUID of the image to be used to represent the face. This will replace previous face image.

Response: application/json
This method returns the updated Face object.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery is not available
`UnknownFaceError`	404	Given face is not available
UnexpectedError	500	An unknown error has occurred while processing the request

PUT `/api/v1/galleries/{galleryId}/faces/{faceId}`
Updates fields of the indicated Face object.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Name	Req.	Type	Description
metadata	yes	JSON object	A new metadata content to replace the one existing in the face object. The metadata will be replaced as a whole.
imageId	yes	UUID	The UUID of the image to be used to represent the face. This will replace previous face image.

Response: application/json
This method returns the updated Face object.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

`UnknownGalleryError`	404	Given gallery is not available
`UnknownFaceError`	404	Given face is not available
UnexpectedError	500	An unknown error has occurred while processing the request

`DELETE` `/api/v1/galleries/{galleryId}/faces/{faceId}`
Deletes the given Face object.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	**Content**
`Authorization`	Bearer ACCESS_TOKEN

Response: Empty response with HTTP status 204.

Errors:

Code	**HTTP Status**	**Message**
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery is not available
`UnknownFaceError`	404	Given face is not available
UnexpectedError	500	An unknown error has occurred while processing the request

B.1.5. FaceBatch Methods

- **Face batches**: Creates a face batch operation. A face batch operation purpose is to upload a package file (ZIP or TAR) containing a large number of images. All of them may be incorporated to the desired gallery, as long as the face extraction and processing is run successfully.

`GET` `/api/v1/galleries/{galleryId}/batches`
Returns a paginated list of available face batches.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Query parameters:

Name	Description
page	Indicates which page of the array will be retrieved.
perPage	Number of items retrieved per page.
search	A search string to constraint retrieved data. This string will look for package names and batch status.
createdAtFrom	A date and time in ISO 8601 format, for filtering only items with a creation date posterior to this value.
createdAtTo	A date and time in ISO 8601 format, for filtering only items with a creation date inferior to this value.

Response: application/json
Returns a paginated list of available FaceBatch objects, with ARRAYNAME=faceBatches.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery is not available
UnexpectedError	500	An unknown error has occurred while processing the request

`POST` `/api/v1/galleries/{galleryId}/batches`
Inserts a batch of faces at the gallery. Faces are contained in a package, supporting at least ZIP, Tar and Gzipped Tar. The operation is performed asynchronously, so it will return an HTTP 202 message containing the identifier to retrieve batch processing status. The status will be updated performing GET calls to the *Location* header of the response.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	Content

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

`Authorization`	Bearer ACCESS_TOKEN

Body:

Name	Req.	Type	Description
packageId	yes	UUID	The package is supposed to contain only face images. The image full path and basename (without extension) will be used to initialize face metadata.

Response: application/json (HTTP status 202)
Returns batch location header and the partially created FaceBatch object.

Headers	Content
`Location`	The URL to poll batch status updates.

Errors:

Code	HTTP Status	Message
ValidationError	400	-
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery is not available
UnexpectedError	500	An unknown error has occurred while processing the request

GET `/api/v1/galleries/{galleryID}/batches/{faceBatchID}`

Returns the face batch object. When the operation is on-going, it will return status code 202 (ACCEPTED) and a status field with PENDING string. Once it is completed, it will return 200. In case of operation failure, you should check the status field which will contain a FAILED indicator, and the completed field which will indicate the number of elements successfully processed.

Request:

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Response: application/json (HTTP status 202 or 200)

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Returns batch location header and the retrieved FaceBatch object (may be partially created).

Headers	Content
`Location`	The URL to poll batch status updates.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery is not available
UnexpectedError	500	An unknown error has occurred while processing the request

B.1.6. Identification Methods

- **Identifications**: Allows to list and create identifications. We say an identification may be synchronous, when the procedure is executed sequentially between the request and the server response. This case is limited to a maximum number of faces in the target gallery, usually 1000. And we say an identification may be asynchronous when it is executed in the background in the server.

`GET` `/api/v1/identifications`
Returns all the identifications related with the given gallery whose ID is passed within the request.

Request:

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Query parameters:

Name	Description
page	Indicates which page of the array will be retrieved.
perPage	Number of items retrieved per page.
search	A search string to constraint retrieved data. This string will look for matches in gallery names, identification status, and rank1 face metadata.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

createdAtFrom	A date and time in ISO 8601 format, for filtering only items with a creation date posterior to this value.
createdAtTo	A date and time in ISO 8601 format, for filtering only items with a creation date inferior to this value.

Response: application/json
Returns a paginated response with Identification objects and ARRAYNAME=identifications.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
UnexpectedError	500	An unknown error has occurred while processing the request

GET `/api/v1/identifications/{identificationId}`
Returns the indicated identification object. Only available for objects with status different than PENDING.

Request:

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Response: application/json (HTTP status 202 or 200)
Returns queried Identification object, with a status 200 in case the identification status is different than PENDING, or 202 in case it is in PENDING status.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownIdentificationError`	404	Unable to locate given identification uuid
UnexpectedError	500	An unknown error has occurred while processing the request

POST `/api/v1/identifications/async`

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Executes an asynchronous identification. It is useful for the identification process over very large galleries, which will require a long time before the identification process is ready. This kind of identification is persisted in the database using a maximum number of candidates (configurable in environmental variables of the container).

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	**Content**
`Authorization`	Bearer ACCESS_TOKEN

Query parameters:

Name	**Description**
page	Indicates which page of the array will be retrieved.
perPage	Number of items retrieved per page.
search	A search string to constraint retrieved data. This string will look for matches in gallery names, identification status, and rank1 face metadata.
createdAtFrom	A date and time in ISO 8601 format, for filtering only items with a creation date posterior to this value.
createdAtTo	A date and time in ISO 8601 format, for filtering only items with a creation date inferior to this value.

Name	**Req.**	**Type**	**Description**
galleryId	yes	UUID	UUID string indicating the target gallery.
imageId	yes	UUID	UUID string indicating the probe image.
minConfidence	no	number	Threshold to filter-out faces which are not considered a match of the identification. By default, it is usually 0.8.
accuracyLevel	no	string	One of: EXACT, HIGH, MEDIUM, LOW. If not given, accuracyLevel will be adjusted depending on the size of the gallery.

Response: application/json (HTTP status 202)

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Returns the partially created Identification object, and the HTTP status is 202 to indicate the object was created, but not fully available. Additionally, a Location header is returned indicating where to retrieve the status of this operation, basically the resource returning the (partially) created Identification object.

Headers	**Content**
`Location`	URI to the Identification object.

Errors:

Code	**HTTP Status**	**Message**
ValidationError	400	-
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery UUID doesn't exists.
`FaceNotFoundError`	415	Unable to locate a face in the image.
`FaceAlignmentError`	415	The face cannot be normalized and aligned.
`MoreThanOneFaceError`	415	More than one relevant faces are present in the image.
UnexpectedError	500	An unknown error has occurred while processing the request

`GET` `/api/v1/identifications/async/{identificationID}`
Returns asynchronous identification results paginated as indicated in query parameters.

Request:

Headers	**Content**
`Authorization`	Bearer ACCESS_TOKEN

Response: application/json
When the identification is pending, it returns a 202 (ACCEPTED) and the PENDING status. After that, and before the identification expires, it returns the list of identification candidates. This endpoint may return all the comparisons performed between the target image (probe) and the gallery. Every time it is called, the identification candidate's expiration will be refreshed. Once it has expired, it will return 404 status. Nevertheless, the identification result will be persisted at `GET /api/v1/identifications/{identificationId}`.

Errors:

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Code	HTTP Status	Message
`PermissionDeniedError`	403	
`UnknownIdentificationError`	404	Given identification UUID has expired or doesn't exists.
UnexpectedError	500	An unknown error has occurred while processing the request

POST `/api/v1/identifications`

Executes synchronous identification. This kind of identification is executed during server operation between request and response. The procedure is run in such a way that any face with a primary key belonging to a different gallery than {galleryId} will be ignored.

Request:

Content-Type: application/json, application/x-www-form-urlencoded

Headers	Content
`Authorization`	Bearer accessToken

Name	Req.	Type	Description
galleryId	yes	UUID	The UUID of the gallery used as target of the identification.
imageId	yes	UUID	The UUID of the image used as probe in the identification.
minConfidence	no	percentage	A threshold to filter-out non-matching candidates.

Response: application/json

Returns the created Identification object.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`EmptyFaceSelectionError`	409	Unable to identify versus an empty set of faces
`FaceNotFoundError`	415	Unable to locate a face in the image.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

`FaceAlignmentError`	415	The face cannot be normalized and aligned.
`MoreThanOneFaceError`	415	More than one relevant faces are present in the image.
UnexpectedError	500	An unknown error has occurred while processing the request

GET `/api/v1/identifications/next-pending`
Returns the next identification object which has not been reviewed by a human agent.

Request:

Headers	**Content**
`Authorization`	Bearer ACCESS_TOKEN

Response: application/json (HTTP status 200 or 204)
Returns an Identification object, with a status 200 in case the identification review state is PENDING, or 204 in case there are no more identifications in PENDING review state.

Errors:

Code	**HTTP Status**	**Message**
`PermissionDeniedError`	403	-
`UnknownIdentificationError`	404	Unable to locate given identification uuid
UnexpectedError	500	An unknown error has occurred while processing the request

POST `/api/v1/identifications/{identificationID}/annotations`
Writes a new Annotation object into the indicated identification operation. Notice that transitions between current decision state and the posted one are validated to be consistent. For instance, a human agent cannot annotate an identification with CONCLUSIVE_* decision to PENDING. But an admin will be allowed to do so.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	**Content**
`Authorization`	Bearer accessToken

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Name	Req.	Type	Description
previousId	yes	UUID	The annotationId of the current Annotation object recorded in the indicated Identification object.
decision	yes	string	One of: PENDING, DONE_CONCLUSIVE, DONE_DOUBTFUL.
observations	no	text	A text to annotate the decision of the human agent.
candidates	no	Array of UUID	An array of faceId of faces marked by the human agent as matches of the identification probe.

Response: application/json
Returns the created Annotation object.

Errors:

Code	HTTP Status	Message
`ValidationError`	400	-
`PermissionDeniedError`	403	-
UnexpectedError	500	An unknown error has occurred while processing the request

B.1.7. Clustering Methods

- **Clusterings**: Allows to create and list asynchronous clusterings on the system. A clustering is an object which allow to put together a bunch of cluster sets. Each cluster is formed by a subset of faces of an indicated gallery, put together depending on similarity between themselves.

`GET /api/v1/clusterings`
Returns all clustering operations, paginated as indicated on query parameters.

Request:

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Query parameters:

Name	Description
page	Indicates which page of the array will be retrieved.
perPage	Number of items retrieved per page.
search	A search string to constraint retrieved data. This string will look for matches in gallery names, clustering status and accuracy level.
createdAtFrom	A date and time in ISO 8601 format, for filtering only items with a creation date posterior to this value.
createdAtTo	A date and time in ISO 8601 format, for filtering only items with a creation date inferior to this value.

Response: application/json
Returns a list of paginated Clustering objects, with ARRAYNAME=clusterings.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
UnexpectedError	500	An unknown error has occurred while processing the request

POST `/api/v1/clusterings`
Performs clustering operation on a given gallery. The operation is performed on background, so it will return almost instantially, but the operation will be enqueued for its execution.

Request:
Content-Type: application/json, application/x-www-form-urlencoded

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Name	Req.	Type	Description
galleryId		UUID	The UUID of the gallery to be clustered.
minConfidence		number	A threshold lower bound to filter-out non

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

			similar faces.
accuracyLevel		string	One of EXACT, HIGH, MEDIUM, LOW, indicating how accurate clusters should be.

Response: application/json (HTTP 202 status)
Returns the corresponding Clustering object and an HTTP 202 status code, indicating the operation has been registered, and the clustering partially created. Additionally, a Location header is returned, indicating the URI to the clustering resource containing the (partially) created object.

Headers	**Content**
`Location`	URI to the location of results for this clustering operation.

Errors:

Code	**HTTP Status**	**Message**
ValidationError	400	-
`PermissionDeniedError`	403	-
`UnknownGalleryError`	404	Given gallery UUID doesn't exists.
UnexpectedError	500	An unknown error has occurred while processing the request

`GET /api/v1/clusterings/{clusteringId}`
Returns the indicated Clustering object.

Request:

Headers	**Content**
`Authorization`	Bearer ACCESS_TOKEN

Response: application/json (HTTP 202 status or HTTP 200 status)
Returns the indicated Clustering object.

Errors:

Code	**HTTP Status**	**Message**
`PermissionDeniedError`	403	-

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

`UnknownClusteringError`	404	Given clustering is not available
UnexpectedError	500	An unknown error has occurred while processing the request

`GET` `/api/v1/clusterings/{clusteringId}/clusters`
Returns the clusters of the indicated Clustering object.

Request:

Headers	**Content**
`Authorization`	Bearer ACCESS_TOKEN

Query parammeters:

Name	**Description**
numFaces__ge	Forces to return clusters with at least a given number of faces. This parameter is useful to filter-out singleton clusters, using a value of 2 here.
page	Indicates which page of the array will be retrieved.
perPage	Number of items retrieved per page.
search	A search string to constraint retrieved data. This string will look for matches in gallery names, clustering status and accuracy level.

Response: application/json (HTTP 202 status or HTTP 200 status)
Returns a paginated list of Cluster objects result as the indicated clustering operation, with the ARRAYNAME=clusters.

Errors:

Code	**HTTP Status**	**Message**
`PermissionDeniedError`	403	-
`UnknownClusteringError`	404	Given clustering is not available
UnexpectedError	500	An unknown error has occurred while processing the request

`GET` `/api/v1/clusterings/{clusteringId}/clusters/{clusterId}`

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Returns the indicated Cluster object.

Request:

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Response: application/json
The indicated Cluster object.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownClusteringError`	404	Given clustering is not available
`UnknownClusterError`	404	Given cluster is not available
UnexpectedError	500	An unknown error has occurred while processing the request

`GET` `/api/v1/clusterings/{clusteringId}/clusters/{clusterId}/faces`
Returns the list of ClusterFace objects that correspond to a given cluster from the indicated clustering operation.

Request:

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Query parameters:

Name	Description
page	Indicates which page of the array will be retrieved.
perPage	Number of items retrieved per page.
search	A search string to constraint retrieved data. This string will look for matches in face metadata.

Response: application/json
Returns a list of paginated ClusterFace objects.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownClusteringError`	404	Given clustering is not available
`UnknownClusterError`	404	Given cluster is not available
UnexpectedError	500	An unknown error has occurred while processing the request

GET `/api/v1/clusterings/{clusteringId}/clusters/{clusterId}/faces/{faceId}`
Returns the indicated ClusterFace object.

Request:

Headers	Content
`Authorization`	Bearer ACCESS_TOKEN

Response: application/json
The response is the content of the indicated ClusterFace object.

Errors:

Code	HTTP Status	Message
`PermissionDeniedError`	403	-
`UnknownClusteringError`	404	Given clustering is not available
`UnknownClusterError`	404	Given cluster is not available
`UnknownFaceError`	404	Given face is not available
UnexpectedError	500	An unknown error has occurred while processing the request

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

B. License

The following clauses set the terms, rights, restrictions and obligations on using this Software, created and owned by VERIDAS DIGITAL AUTHENTICATION SOLUTIONS, S.L. ("VERIDAS" or 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.

1. LICENSE GRANT

VERIDAS hereby grants to the Licensee a non-exclusive, non-assignable and non-transferable, non-commercial, indivisible, without the rights to create derivative works license for the term specified in the Offer to use the offered software (the "Software") for the specific purpose specified between the Parties and/or in the Offer, 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, as may be described in the Software. Even if such components are governed by other agreements, the disclaimers and the limitations on and exclusions of damages below also apply.

On specific products, if necessary, VERIDAS may install a computer application, in some cases connected with an external server, that allows VERIDAS to verify that the system is updated and payments are correctly made.

2. USE OF THE SOFTWARE

2.1. The Licensee cannot use the Software for other purposes than as specified in the Offer.

2.2. The Licensee may permit its employees to use the Software for the purposes agreed by the parties and/or described in the Offer, provided that the Licensee takes all necessary steps and imposes the necessary conditions to ensure that all employees using the Software do not commercialize or disclose the contents of it to any third party, or use it other than in accordance with the terms herein.

2.3. The Licensee will not distribute, sell, license or sub-license, lease, trade or expose for sale the Software to a third party.

2.4. No copies of the Software are to be made other than as expressly approved by VERIDAS.

2.5. No changes to the Software or its content may be made by Licensee.

2.6. The Licensee will provide technological and security measures to ensure that the Software, which the Licensee is responsible for, is physically and electronically secure from unauthorized use or access.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

2.7. The Licensee shall ensure that the Software retains all VERIDAS copyright notices and other proprietary legends and all trademarks or services marks of VERIDAS, as specified in clause 3 below.
2.8. The Licensee shall not, under any circumstances, use reverse engineering practices on the Software.
2.9. The Licensee is responsible for extending the obligations herein to Clients, to the extent they may apply.

3. INTELLECTUAL PROPERTY RIGHTS

3.1. Intellectual Property Rights means all rights in and to any copyright, trademark, trading name, design, patent, know-how, trade secrets and all other rights resulting from intellectual activity in the industrial, scientific, literary or artistic field and any application or right to apply for registration of any of these rights and any right to protect or enforce any of these rights.
3.2. All Intellectual Property Rights over and in respect of the Software are owned by VERIDAS. The Licensee does not acquire any rights of ownership in the Software, and it must use the Intellectual Property Rights exclusively as required for reasonable and customary use within the purposes of the License.
3.3. Any modification made on the Software in order to adapt it for the provision of the service to the Licensee and/or the Client, shall be included in the Intellectual Property Rights as defined in clause 3.2 above.

4. LIMITATION OF LIABILITY

4.1. To the extent permitted under the law, the Software is provided under an “AS IS” basis. The Licensee acknowledges and agrees that neither VERIDAS nor its board members, employees or agents, will be liable for any lost or damage arising out of or resulting from VERIDAS’ provision of the Software under this License, or any use of the Software by the Licensee, the Client or their employees. The Licensee hereby releases VERIDAS to the fullest extent from any such liability, loss, damage or claim, both its own or of the Clients. Regarding the processing of personal data with the Software, the Licensee acknowledges and agrees that the Licensor is not liable for the data collected by the use of the Software within the scope of the Licensee’s activities.
This limitation shall apply to any issue related to the software, services, contents (including code) found at third-party websites or third-party programs.
4.2. 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.

4.3. Notwithstanding the previous general clause, VERIDAS expressly excludes any liability resulting from:

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.

a) willful, fraudulent, deliberately unlawful acts, penalized as a criminal offence, or which are voluntarily against the law, carried out by or against the Licensee or a Client, or their employees;
b) any fact or circumstance, real or suspected, the Licensee or the Clients know about or could have reasonably foreseen, and that may affect, in any way, to the correct functionality of the Software;
c) mechanical fails, electric fails (including interruptions, outages, overvoltages or power cuts) and fails on telecommunication or satellite transmission systems, given that those fails are not due to an act or omission of VERIDAS or to an error of the delivered Software;
d) damage or loss of the Licensee's or Client's data stored or hold in VERIDAS' systems, as well as the costs resulting from such circumstances; or
e) any act or claim alleging, derived from or based on funds, money or value transfers or any other negotiable instrument for or from a bank or financial institution.

4.4. Licensee and Clients acknowledge that the Software has an associated error rate which makes not possible, considering the state of the art, to guarantee a complete level of reliability. In this sense, VERIDAS provides information about the levels and error rates of every Software version. On the other hand, the outputs of the system are not binary but they offer a probabilistic result that has to be configured by the Licensee or the Client.

The facial verification functionality of this Software has been tested against the LFW database, with results up to a 99.8% precision (FNR=0.4% at FPR≤0.4%). 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.

As a result, it is the responsibility of the Licensee and/or the Client to carry out any necessary internal control evaluation, to implement due diligence measures in accordance with the regulation they have to comply with (including without limitation: regulation on the prevention of money laundering, citizen security, border controls, etc.) and to evaluate the convenience of the Software as an instrument for complying with current legislation.

VERIDAS provides some tools that may help the Licensee and/or the Client to implement prevention mechanisms, but it is not responsible for the study of the convenience of its implementation, the specific configuration of the Software, or the use of the Software (expressly including the validation results obtained with the Software).

4.5. Licensee is responsible for communicating and transferring the previous limitations of liability to the Clients.

All rights reserved – This document contains confidential information, property of Veridas Digital Authentication Solutions, S.L., and cannot be reproduced, copied, or revealed to third parties, without the express written authorization of Veridas. The information of this document must be kept secret and used in the exclusive benefit of Veridas.