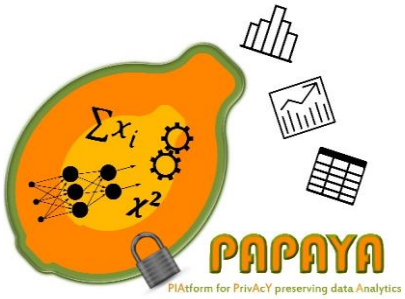


D5.4 – PAPAYA PLATFORM GUIDE

Work Package	WP 5, Platform Validation
Lead Author	Boris Rozenberg, Ron Shmelkin (IBM)
Contributing Author(s)	Beyza Bozdemir, Orhan Ermis, Melek Önen (EURC) Sebastien Canard, Bastien Vialla (ORA) Angel Palomares Perez, Nuria Ituarte (ATOS) Tobias Pulls, Simone Fischer-Hübner, Tobias Vehkajärvi (KAU) Eleonora Ciceri, Marco Mosconi (MCI)
Reviewers	Nuria Ituarte (ATOS) Tobias Pulls (KAU)
Due date	30.04.2021
Date	30.04.2021
Version	1.0
Dissemination Level	PU (Public)

The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Programme, through the PAPAYA project, under Grant Agreement No. 786767. The content and results of this deliverable reflect the view of the consortium only. The Research Executive Agency is not responsible for any use that may be made of the information it contains.

Project No. 786767

Revision History

Revision	Date	Editor	Notes
0.1	2.03.2021	Boris Rozenberg (IBM)	ToC
0.2	8.04.2021	All contributing authors	A version for the 1st internal review
0.3	22.04.2021	All contributing authors	A version after the 1st internal review
0.4	28.04.2021	All contributing authors	A version after the 2nd internal review
1.0	30.04.2021	Orhan Ermis, Melek Önen	Quality check

Project No. 786767

Table of Contents

Project No. 786767

Executive Summary

Rather than several standalone modules, the PAPAYA project aims at developing an integrated platform for privacy-preserving data analytics to make them available in a broad spectrum of products and services, with usable, friendly and accessible safeguards options. The main goal of the platform is to be used by service developers to deploy and run privacy-preserving services, and by service consumers, that are interested to employ privacy-preserving analytics. In the deliverables D4.1 [1], D4.2 [2], and D4.3 [3] we presented the platform functional design, architecture, and explained how different privacy-preserving primitives developed in WP3 (see D3.1 [4] and D3.3 [5]) are integrated into the platform in a way that they will be interoperable/compatible with each other and could work together in the integrated platform. We also presented the design of platform dashboards that provide the UI, configuration functionality, and visualization functionality.

In this deliverable we summarize all functionality and workflows of the PAPAYA platform and provide a detailed description of how the users should interact with the system, including deployment instructions and API for each platform service. This deliverable is an outcome of the Task T5.3 (i.e., “Technology assessment and recommendation”), and its integration with the Deliverable D5.3 (i.e., “Refinement recommendations for the platform”) serves as final validation of the PAPAYA platform. Indeed, these two documents, when combined, describe the current implementation and usage of the PAPAYA services, tools, and dashboards, and provide refinements for possible future expansions.

As already mentioned in D4.1 [1], PAPAYA platform services are specified based on the four generic usage scenarios, namely upload model, create model, apply model and collaborative training. In upload model, an already trained machine learning (ML) model can be uploaded to the PAPAYA platform when the client wants to delegate the computationally intensive task (which is applying a model on the client’s sensitive data) in a privacy-preserving manner. The create model is used when the client is not able to create the ML model; therefore, the PAPAYA platform generates a model on the protected data shared by the client. Apply model is the use case where already uploaded or created model is applied on the client’s protected data in a privacy-preserving manner. Finally, in collaborative training, two or more participants perform a ML training collaboratively while preserving the privacy of the training data.

The PAPAYA platform consists of the following groups of services:

- Privacy-preserving analytics defined in the deliverables D3.1 [4] and D3.3 [6] such as classification on Neural Networks, privacy-preserving clustering, privacy-preserving statistics, and privacy-preserving collaborative training of Neural Networks
- Security and transparency services including the identity access management (IAM) for authentication and authorization services to the different components that will be integrated in the PAPAYA platform, auditing to support auditing and towards being able to hold stakeholders accountable for their use of PAPAYA, and key manager for managing the cryptographic material during the whole lifecycle of the PAPAYA project in the cases where it will be required.

- The PAPAYA platform provides two dashboards for configuration and visualization, namely the platform dashboard and the agent dashboard. The platform dashboard is used for configuration and monitoring the services provided by the platform whereas the agent dashboard is used for viewing the data processing logs from an agent and for showing the configuration of the agent.
- The data subject toolbox provides a number of mostly independent tools (Explaining Privacy-preserving Analytics, Data Disclosure Visualization, Annotated Log View and Privacy Engine) which focus on different privacy and data protection issues surrounding privacy-preserving data analytics. The data subject toolbox enables a platform user to build an integrated data subject dashboard as part of their data subject facing software, e.g., as part of a mobile app.

As a proof of concept usage, the platform is deployed on IBM Kubernetes cloud service. In addition to that, all services are designed to be generic to be deployed on any other cloud platforms.

Glossary of Terms

2PC	Two-Party Computation
AA	Auditing Agent
ABY	Arithmetic sharing, Boolean sharing and Yao's garbled circuits framework
AC	Auditing Collector
ALT	Annotated Log view Tool
API	Application Programming Interface
BF	Bloom Filters
BGV	Brakerski-Gentry-Vaikuntanathan homomorphic encryption scheme [7]
BFV	Brakerski/Fan-Vercauteren fully homomorphic encryption scheme [8, 9]
CA	Certificate Authority
CKKS	Cheon-Kim-Kim-Song fully homomorphic encryption scheme [10]
CLI	Command Line Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CR	Container Registry
DB	Data Base
DC	Data Controller
DP	Differential Privacy
DNN	Deep Neural Network
DS	Data Subject
DSRM	Data Subject Rights Manager
DVT	Disclosure Visualization Tool
ECG	Electro cardiogram
ES	ElasticSearch
FC	Fully Connected
FE	Functional Encryption
FHE	Fully Homomorphic Encryption
GRU	Gated Recurrent Unit
HE	Homomorphic Encryption
IAM	Identity Access Manager
KM	Key Manager
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Network
PE	Privacy Engine
PHE	Partially Homomorphic Encryption
PoC	Proof of Concept
PPM	Privacy Preferences Manager
RAM	Random Access Memory

REST	Representational State Transfer
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SIMD	Single Instruction Multiple Data

1 Introduction

1.1 Purpose and Scope

The purpose of this deliverable is to provide an overview of all the services available in the PAPAYA platform and provide a comprehensive description of how these services could be deployed and used by PAPAYA clients. The intended audience for this document consists of two main groups: (1) service developers, who could use the document to understand how to deploy new services on the PAPAYA platform; and (2) service users, who could use the document to figure out what services are available on the PAPAYA platform, and how to incorporate them in their applications.

1.2 Structure of the Document

The rest of the document is organized as follows:

- **Section 2** provides a high-level overview of the PAPAYA framework, including description of main stakeholders.
- **Section 3** provides details on how to deploy the platform and platform services, and presents PAPAYA dashboards, namely platform dashboard and agent dashboard.
- **Section 3** provides an overview of the core PAPAYA services and describes in detail how they can be deployed and used. The core services are defined in the deliverable D3.1 [4] and deliverable D3.3 [6]. Provided services in the scope of this deliverable are: (1) Privacy-preserving classification on Neural Networks; (2) Privacy-preserving collaborative training of Neural Networks; and (3) Privacy-preserving clustering.
- **Section 5** gives details about how security and transparency is achieved in the platform, including authentication, authorization, auditing and key management for cryptographic tools (if it is required).
- **Section 6** dedicated to Data Subject Toolbox, which provides versatile tools (Explaining Privacy-preserving Analytics, Data Disclosure Visualization, Annotated Log View and Privacy Engine) related to the data subject privacy. Tools presented in this section can then be used to form a *data subject dashboard*.
- We conclude the deliverable in **Section** 7.

2 PAPAYA Framework

2.1 Main Stakeholders

On a high-level, as mentioned in the deliverable D4.2 [2], there are four main stakeholders in the PAPAYA framework:

1. **Platform clients*:*** stakeholders who wish to perform some analytics in a privacy-preserving manner. **Platform clients** can be considered as Data Controllers or external queriers who are allowed to request some analytics results while not being the actual owners of the data.
2. **Platform administrators*:*** responsible for platform administration purposes such as resource allocation or monitoring.
3. **Service providers**: the author of the services available on the platform.
4. **Data Subjects: end-users (**e.g. application user**)** of the **Platform clients**.

Details about PAPAYA usage scenarios can be found in D4.1 [1].

2.2 PAPAYA framework architecture

The PAPAYA framework (see Figure 1) is composed of two main groups of components: (1) the platform-side components that run on the (non-trusted, but semi honest) Kubernetes[1] cloud server (see Figure 1, PAPAYA Platform Kubernetes Service); and (2) client side components, that run on trusted client environment (see Figure 1, Client Side).

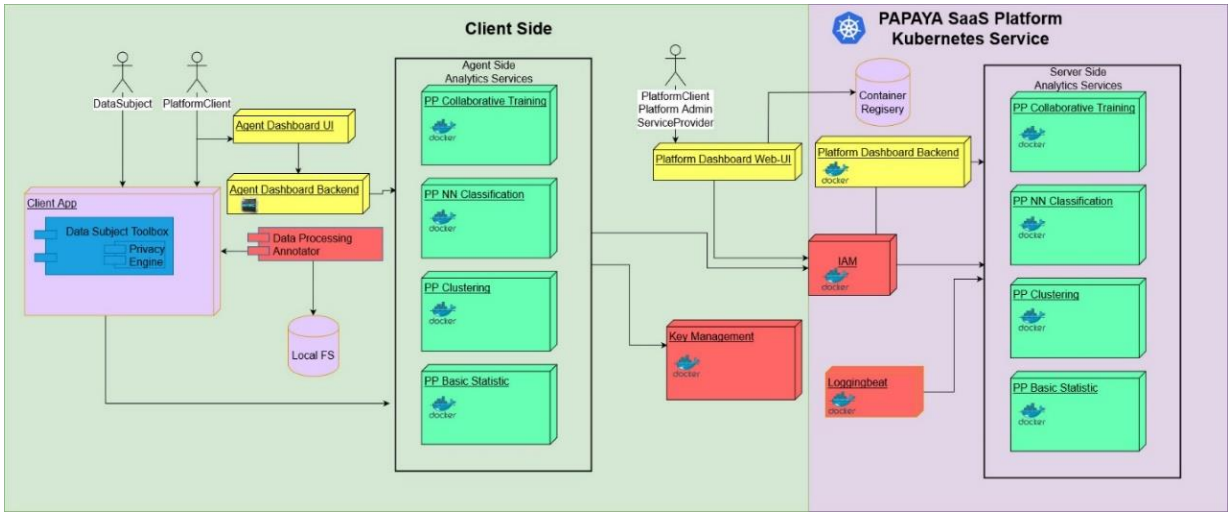


Figure 1 PAPAYA Framework architecture

[1] https://kubernetes.io/

From the functional perspective, the PAPAYA framework components can be regrouped into the following categories (more details about each category can be found in deliverables D4.1 [1] and D3.1 [4]):

- **The privacy-preserving analytics services** (colored with green in Figure 1) which allow platform clients to perform analytics of interest, in a privacy-preserving manner. We support the following analytics:
  1. Privacy-preserving NN classification. We provide four services for applying neural network for the purpose of classification in a privacy-preserving manner: (1) 2PC-based; (2) PHE-based; (3) FHE-based; and (4) Hybrid approach. Each one could be preferable than others in different settings, mainly depending on NN architecture
  2. Privacy-preserving collaborative training of NN. The service allows multiple participants to perform a ML training collaboratively, while preserving the privacy of the training data.
  3. Privacy-preserving trajectory clustering. The service provides means to cluster trajectories in a privacy-preserving manner.
  4. Privacy-preserving basic statistics. The service provides means for privacy-preserving computation of statistics using functional encryption and privacy-preserving counting using Bloom Filters.

  The users of these services are platform clients who can either be Data Controllers or external queriers (in the case for privacy-preserving trajectory clustering who are authorized by Data Controllers to request some analytics results. Each service is divided into two parts:
  1. Server – responsible for performing analytics of interest on encrypted data and will run on a PAPAYA's Kubernetes cluster.
  2. Agent – responsible for communication with the appropriate server-side component and responsible for managing cryptographic operations for the client. The agent will be downloaded from the Container Registry (CR) as a Docker image and deployed on a client side. Deployment and execution is under the responsibility of the platform client.
- **The platform security and transparency services** (colored with red in Figure 1) provide platform authorization, authentication (Identity and Access Management - IAM), auditing and cryptographic Key Manager (if needed).

- **The PAPAYA dashboards** (colored with yellow in Figure 1):

  1. **Platform dashboard** allows: (1) **service providers** to deploy privacy-preserving services; (2) **platform clients** to choose/run privacy-preserving services and review operational; and (3) **platform administrators** to configure and manage the platform and review auditing logs.
  2. **Agent dashboard** allows **platform clients** to visualize the configuration of the agent running on the client side and review operational and auditing logs.

- **Data Subject Toolbox** (colored with blue in Figure 1) consists of a number of mostly independent tools (DS Tool 1: Explaining Privacy-preserving Analytic, DS Tool 2: Data Disclosure Visualization, DS Tool 3: Annotated Log View and DS Tool 4: Privacy Engine), which provide versatile tools for data protection by design by platform clients (acting as data controllers) towards data subjects whose personal data is processed in their services.

3 PAPAYA Platform Deployment

The architecture of the PAPAYA platform was designed with the intent to allow easy deployment and easy integration with complementary components in order to provide a complete solution for privacy-preserving analytics. The addition of new privacy-preserving analytics can be done without much effort. The deployment of server-side instance of any analytics is performed seamlessly on the client's demand. Even though the PAPAYA platform presents the Platform as a service approach, the service implementation designed with the intent to allow easy adoption of each of the services as a stand-alone service (SaaS).

3.1 Platform initialization

For the proof of concept (PoC) usage, the platform is deployed on IBM's cloud service, however its design is platform agnostic and can be easily deployed on any other K8s cluster provider.
The platform administrator (IBM) creates an account of K8s service and allocates the required resources for the Kubernetes cluster. In addition, the platform administrator allocates the following required external services:

1. Container Registry (CR). The platform administrator creates a single namespace (**papaya-de)**.
2. The platform admin creates a main dedicated K8s namespace, all analytics services will be deployed within this namespace (**papaya).**
3. Permissions Provided to the platform users
4. Creates K8s Service Account with admin privileges, this service account issued by the platform dashboard. This is essential since the platform dashboard will automatically deploy instances of analytics services.
5. Creates Elasticsearch and Kibana internal instances.

The platform administrator creates the following permission groups:
1. Platform administrator
    a. Allocate resources on the Kubernetes cluster
    b. Allocate/Create external services
    c. Admin access to CR
2. Service providers
    a. Active access to the CR
    b. If the service requires integration/communication with an external service, appropriate accounts should be created or allocated.
3. Platform Clients
    a. Should be able to download only the agent side container from the CR

To support auditing, the platform administrator deploys Logstash, Filebeat as deamonset on the Kubernetes cluster, Elasticsearch instance, and Kibana instance for logs presentation and auditing.

3.2 Platform dashboard deployment

Platform dashboard is implemented as a Flask[2] web application hosted in a Docker container.
To store the required information, the platform dashboard uses SQLite database. The database instance is stored in K8's Persistent Volume.
The platform administrator will deploy the platform dashboard on the K8s, using YAML files (see Appendix 1). The platform dashboard deployed as a K8s service that uses Ingress[3] to allow secure connection.

The dashboard provides the following functionality:

1. Add a new user
2. Present a list of services provided by the platform (Service's Catalog)
3. Add/Edit/Delete services
4. Create/Deploy/Delete application (a dedicated instance of each of the provided services)
    a. The platform supports deployment of application with following communication setups:
        i. HTTP
        ii. Socker
        iii. HTTP + Socket
    b. Services with HTTP communication channel, can be easily integrated with IAM (for more details see Section 5.1)
5. Allows application's owners to monitor the flow of the application by presenting operational logs.
6. Allows admin users to review all instances' operational logs, via integrating the Kibana dashboard into the platform admin's log view.
7. Allows logs auditing by observing logs authenticity.

3.3 Service deployment

This section describes the steps needed in order to upload or use a service. For the PoC usage we use a Container Registry Provided by IBM Cloud. The steps are divided into two phases: (1) uploading phase; and (2) usage-execution phase.
The following steps are common for the both phases:

1. Install the IBM Cloud CLI[4]
2. Install the Docker CLI [5]
3. Log in to your IBM Cloud account.
   `ibmcloud login -a cloud.ibm.com -r eu-de -g papaya`

[2] https://flask.palletsprojects.com/en/1.1.x/
[3] https://kubernetes.io/docs/concepts/services-networking/ingress/
[4] https://cloud.ibm.com/docs/cli?topic=cloud-cli-getting-started#idt-prereq
[5] https://docs.docker.com/install/

The uploading phase is performed by the service provider user and it consists of the following steps:

1. Log your local Docker daemon into the IBM Cloud CR:
   ibmcloud cr login
2. Create two images: server side and client-side agent.
   The server-side container will run on papaya's K8s cluster. The client-side agent will run on the client's side. Instructions on how to create the Docker image can be found in the following link:
   https://docs.docker.com/develop/develop-images/baseimages/
3. Upload each image to the appropriate namespace on the CR.
   for the service images use:
   docker tag <local container name> de.icr.io/papaya-de/<my_repository>:<my_tag>
   docker push de.icr.io/papaya-de/<my_repository>:<my_tag>
   More details and CR commands can be found in Kubernetes documentation[6].

4. The service provider user will add a new service to the services catalog on the Platform dashboard[7] (see Figure 2), while specifying all required fields and describing in details the service's flow and requirements (see Figure 3).

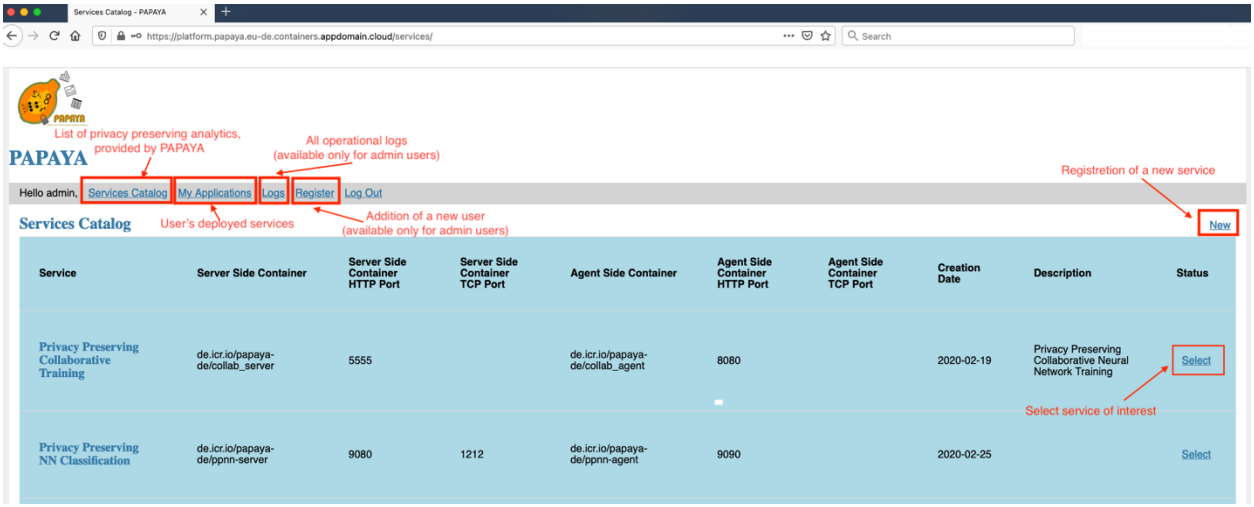


Figure 2: Platform Dashboard -Services Catalog

[6] https://cloud.ibm.com/kubernetes/registry/main/start
[7] https://platform.papaya.eu-de.containers.appdomain.cloud

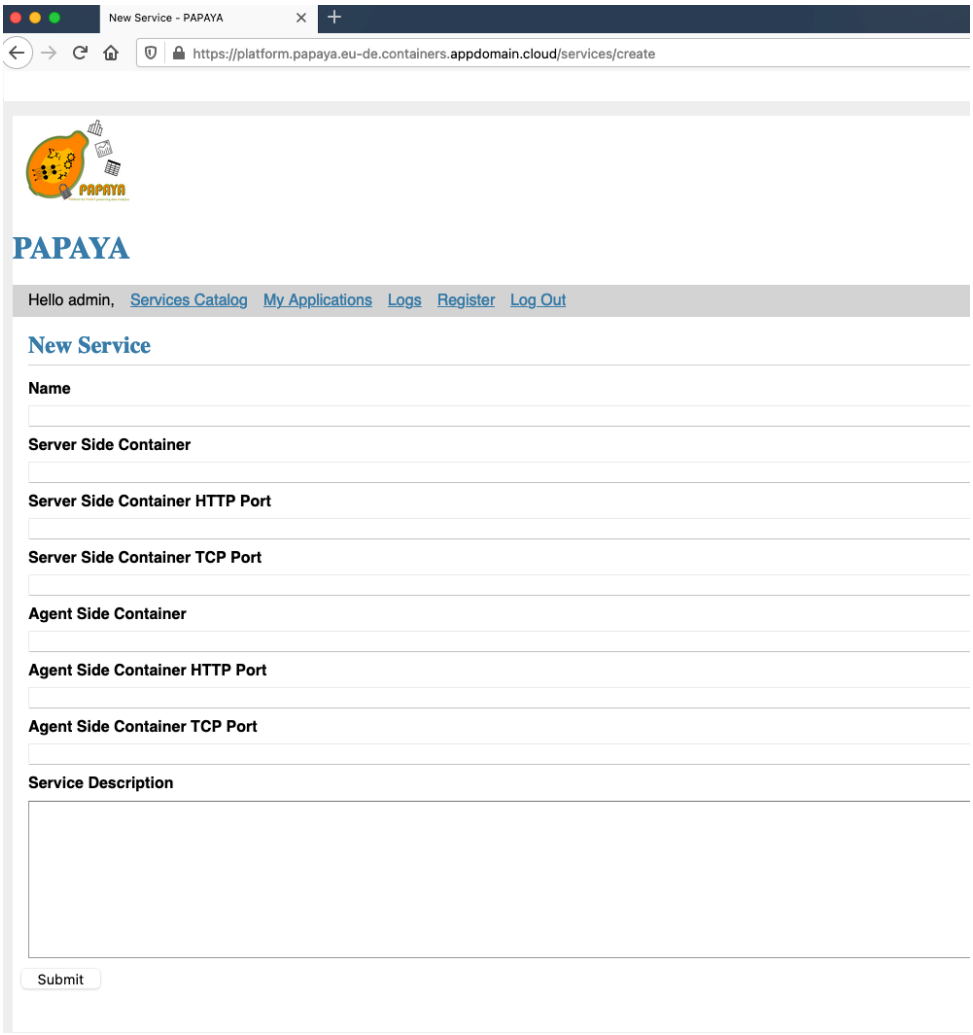


Figure 3:Platform Dashboard - New Service Registration

3.4 Agent and Agent Dashboard

3.4.1 Agent deployment

The usage-execution flow consists of the following steps:

1. The platform client will login to the Platform dashboard[8].
2. The client will select the tab of services catalog, choose the service of interest (see Figure 2), and create a new instance of the desired service (see Figure 4)
3. The client will download the client-side agent image by using the following commands:

[8] https://platform.papaya.eu-de.containers.appdomain.cloud/auth/

```
ibmcloud cr login
docker pull <image name>
```

where the <image name> is presented in the “services catalog” view on a platform dashboard under the “Agent Side Container” (see Figure 2) name and will be structured of the following:

```
de.icr.io/papaya-de/<image>:<tag>
```

4. After the services are deployed on the k8s cluster (by the platform dashboard application), the platform will present the URL of the deployed service for the HTTPS communication or provide a cluster IP and port for TCP (socket) communication (see Figure 5), For example:

```
SERVER_URL=https://29dr4d.papaya.eu-de.containers.appdomain.cloud
SERVER_TCP_PORT=32067
SERVER_IP=158.177.182.219
```

In addition, the platform dashboard provides the option to download the file (env file) that will contain these parameters (accordingly to the communication protocol with the service). The client administrator should be able to provide this file to the client agent, based on how the agent expects to receive these parameters. Possible ways of doing that are as follows:

a. Provide the env file as env. variables for the client agent execution.

```
docker run with an --env-file <file name>
```

b. Read the env file by the client’s application and provide it to the agent side within the initiation step. Meaning that the agent side will provide INIT API and will expect to receive all essential parameters to communicate with appropriate server side.

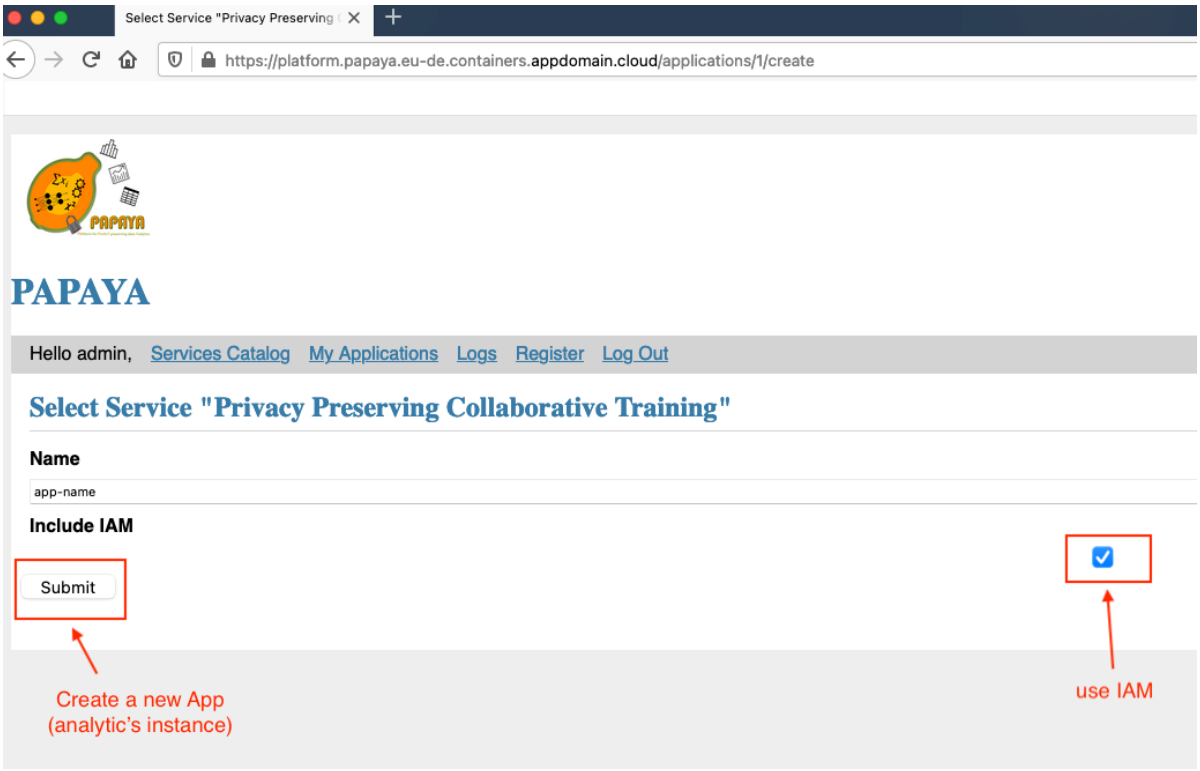


Figure 4: New instance creation

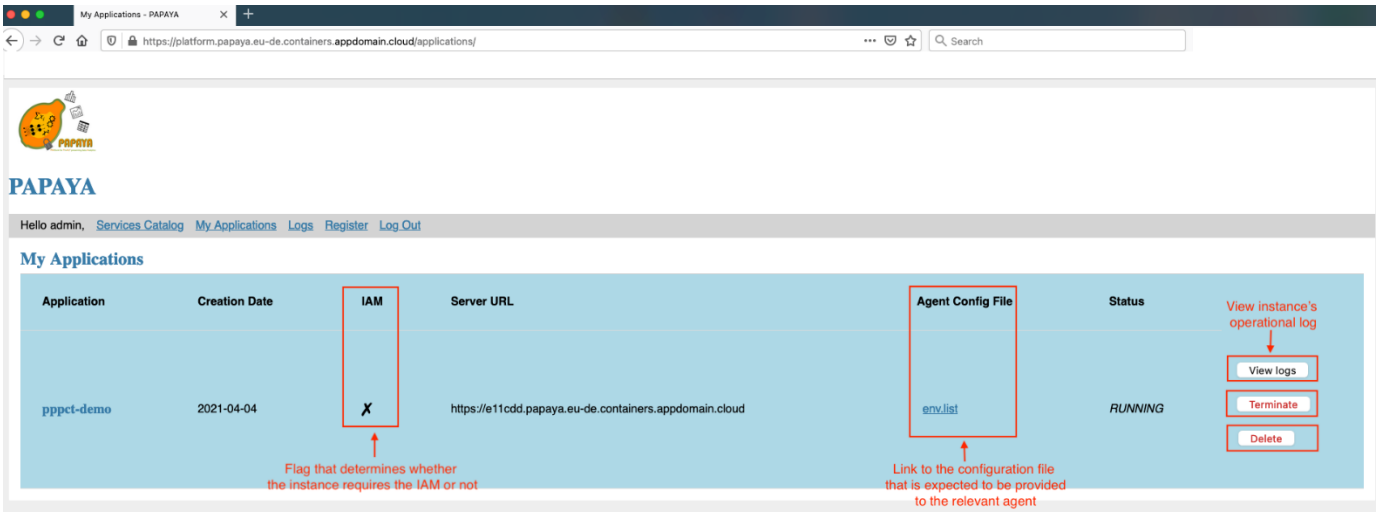


Figure 5: My List of Applications

3.4.2 Agent dashboard

The Agent Dashboard targets the development phase of when a new user of the PAPAYA platform is working on integrating a platform agent into its existing systems. The target user of the Agent Dashboard is therefore a developer that has just downloaded an agent from the platform.

Next, we provide an overview of the Agent Dashboard, using the collaborative training agent described in Section **Error! Reference source not found.** as the example.

First, using a shell script, the developer runs the collaborative training agent:

```
user:~/PAPAYA/agentdashboard $ ./start_collab_agent.sh
d442d0b1f9b439a1a0ce1711abf91152020f8d0f2b203e183babbcdd83705a21
user:~/PAPAYA/agentdashboard $
```

Next, the developer inspects the logs with the help of the Agent Dashboard:

```
user:~/PAPAYA/agentdashboard $ ./ad logs collab
Showing logs for collab...
user:~/PAPAYA/agentdashboard $
```

This opens up the developer's default browser with the logs, as shown in Figure 6.

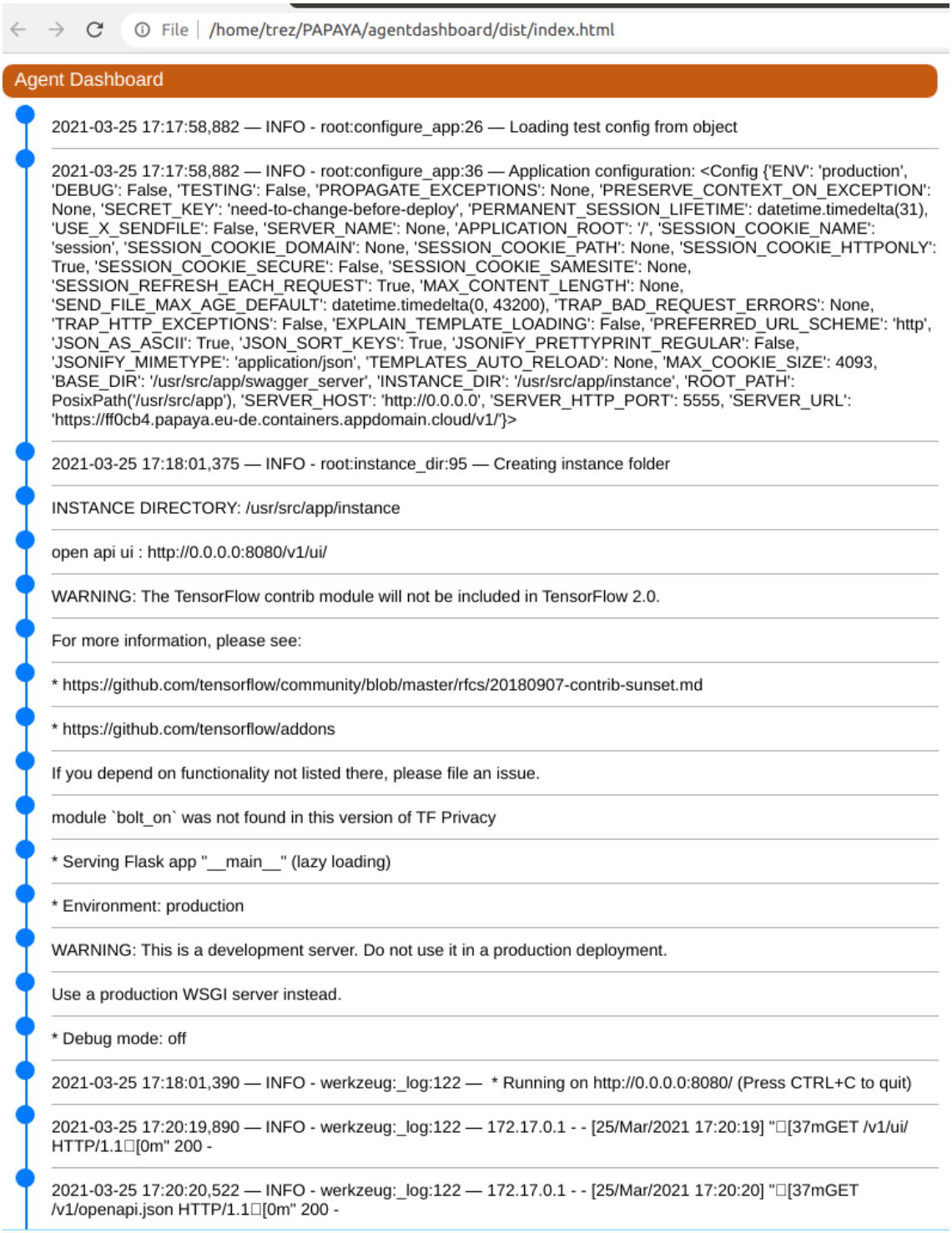


Figure 6: Agent Dashboard view of logs.

Project No. 786767

To better understand how the agent works, the developer can view its configuration:

```
user:~/PAPAYA/agentdashboard $ ./ad explain collab
Showing details for collab
Configuration
{
  "ct": {
    "apply_dp": false,
    "download_fraction": 1,
    "download_frequency": 1,
    "upload_fraction": 0.5,
    "upload_frequency": 1,
    "waiting_time": 10
  },
  "record_level_dp": {
    "l2_norm_clip": 1,
    "learning_rate": 0,
    "loss": "string",
    "metrics": [
      "string"
    ],
    "microbatches": 0,
    "noise_multiplier": 1.1,
    "optimizer": "sgd"
  },
  "user_level_dp": {
    "budget_per_gradient": 1,
    "gamma": 0.01,
    "sensitivity": 0.05
  }
}
user:~/PAPAYA/agentdashboard $
```

Beyond showing the JSON configuration of the agent in the terminal, the developer's browser is once again opened, this time showing the interactive explanation of the agent, see Figure 7.

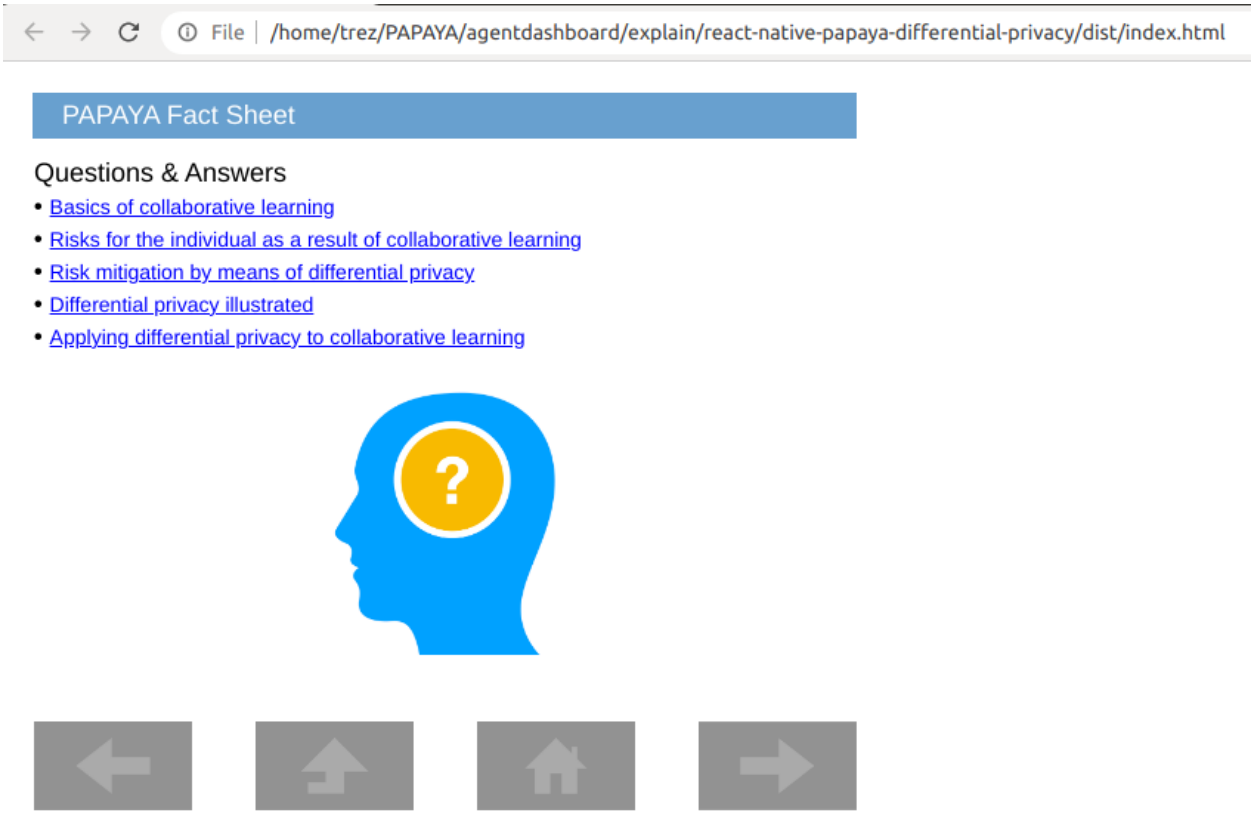


Figure 7: Explanation of an analytic as part of the Agent Dashboard.

The explanation is based on the components of the Data Subject Toolbox, see Section **Error! Reference source not found.**. The Agent Dashboard is available at https://git.cs.kau.se/papaya/agentdashboard.

4 Platform Core Services

4.1 Apply Neural Network Model

In this section, we describe several services for applying neural network for the purpose of classification in a privacy-preserving manner. The detailed description of the underlying technology realized by these services can be found in deliverables D3.1 [4] and D3.3 [6].

Prior to using any of the services described in this section, a Neural Network model should be trained locally based on the clear text data with all the required optimization and the dimensionality reduction. After achieving the desired accuracy, the trained model (i.e. architecture and weights) should be saved in a supported format and passed to the service later as described in each of the following subsections.

4.1.1 Privacy-preserving NN classification based on 2PC

4.1.1.1 Overview and main functionality

This solution provides a privacy-preserving NN classification for arrhythmia detection using secure two-party computation (2PC). The service has two main components as described in D4.1 [1], namely the server-side and client-side components. Those components are deployed as Docker containers to the PAPAYA platform as presented in the deliverable D4.2 [2]. In short, the server-side component provides the NN parameters specifically designed for the arrhythmia classification and the client-side component provides the input (ECG signals) to be processed. Later, once the inputs are provided for the classification, the client-side and server-side component execute 2PC NN classification. As described in deliverables D4.1 [1] and D4.2 [2], this service uses both HTTP connection and TCP connection while executing the classification. Finally, we provide details about how we evaluate the integration of the service in the deliverable D4.2 [2].

4.1.1.2 Deployment

The service is deployed by following the instructions in Section **Error! Reference source not found.** (i.e. Service deployment) and Section **Error! Reference source not found.** (i.e. Agent deployment).

4.1.1.3 How to use/API

In this section we describe the service API, swagger descriptions are available in Appendix 3.1.1.

Server-side component API:
There is only one RESTful API call at the server-side component in this service and this for initiating the classification for the server side (*classify/*). The rest of the communication between

the server-side component and the client-side component is realized by using ABY[9] sockets. This call is invoked by the client-side component when the classify/ API (see client-side component APIs section) of that component is called.

Client-side component API:
There are two RESTful API calls at the client-side component, namely *init* and *classify*. The rest of the communication between the server-side component and the client-side component is realized using ABY sockets.

The *init* API call is used for initializing the public IP address and the port number of the server. Following is an example of the call:
curl
http://[address_of_the_client]/init/[IP_of_the_server]/[url_of_the_server]/[port_of_the_server]/

The *classify* API call is used for initiating the classification at the client side. Following is an example of the call:
curl -F 'file=@[name_of_the_input_file]' http://[address_of_the_client]/classify/

4.1.2 Privacy-preserving NN classification based on PHE

4.1.2.1 Overview and main functionality

With this solution based on the Paillier partially Homomorphic Encryption (PHE) scheme, a party can classify any personal input (an image or an ECG input) by using a confidential pre-trained NN stored on the server-side. This service consists of two main components: the client-side components and the server-side components previously presented in D4.1 [1]. Both of them are realized as Docker containers and deployed on the PAPAYA platform. In this solution, the pre-trained NN is held by the server, and the client, who would like to receive the result of the inference, encrypts its input with its public key. Then, the encrypted input is sent to the server. The server-side runs the PHE NN classification over the encrypted input until the square layer. When the PHE NN classification comes to square, the server-side interacts with the client-side to perform the square function via socket programming. After the square computation is done, the server-side continues the PHE NN classification until the next square layer (if exists). As described in deliverables D4.1 [1] and D4.2 [2], this service uses HTTP connection and TCP connection while executing the classification. Finally, we provide details about how we evaluate the integration of the service in the deliverable D4.2 [2].

4.1.2.2 Deployment

The service is deployed using the instructions detailed in Section **Error! Reference source not found.** (i.e. Service deployment) and Section **Error! Reference source not found.** (i.e. Agent deployment).

[9] https://github.com/encryptogroup/ABY

4.1.2.3 How to use/API

In this section we describe the service API, swagger descriptions are available in Appendix 3.1.2.

Server-side component API:
There is only one REST-full API call for the server-side component in this service, namely "*classify*/". This API call is used for initiating the classification. Once the classification is initiated, then all necessary communications for executing classification are accomplished through socket calls between the server-side and client-side components. Similar to 2PC base NN Classification service introduced in Section **Error! Reference source not found.**, server-side component API is automatically invoked by the client-side component when this service's classify API call is invoked.

Client-side component API:
There are two REST-full API calls at the client-side component, namely *init* and *classify.* Socket calls are used for communication purposes between the server-side and client-side components.

The *init* API call is used for initializing the public IP address and the port number of the server. Following is an example of the call:
curl
http://[address_of_the_client]/init/[IP_of_the_server]/[url_of_the_server]/[port_of_the_server]/

The *classify* API call is used to initiate the classification at the client side. Following is an example of the call:
curl -F 'file=@[name_of_the_input_file]' http://[address_of_the_client]/classify/

4.1.3 Solution based on Homomorphic Encryption

4.1.3.1 Overview and main functionality

This solution uses homomorphic encryption to provides a privacy-preserving service that perform the inference of neural network. The encryption is handled by SEAL library [11]. The service is composed of two components, one for the client and one running the computations on the server. The solution aims mostly at dense or convolutional network with small depth. Also, the solution gives the possibility to evaluate the model on multiple data in parallel. The details of the design, deployment and integration are given in D4.1 [1], D4.2 [2] and D4.3 [12]. The details of the performance can be found in D3.3 [6].

4.1.3.2 Deployment

To deploy the service, follow the instructions in Section 3.3 (Service deployment) and Section 3.4.1 (Agent deployment).

4.1.3.3 How to use/API

In this section we describe the service API, swagger descriptions are available in Appendix 3.1.3.

Client-side component API:

The *init* call is used to initialize the service by sending the model architecture, weights, and the encryption context to the server-side component. Following is an example of the *init* API call:

```
curl -F "NN_description=@[archi_file]"
     -F "NN_weights=@[weigths_file]"
     -F "HE_context=@[context_file]"
     http://[address_of_the_client]/Init/
```

With the *keygen* call the component will generate the keys for the encryption scheme. Following is an example of the *keygen* API call:

```
curl http://[address_of_the_client]/Keygen/
```

The *classify* call is used to classify data. The call takes the data as an input, encrypt it, and send it to the server to be classified. When the encrypted result is received, it is decrypted, and the result is provided to the caller. Following is an example of the classify API call:

```
curl -F "data=@[data_file]"
     http://[address_of_the_client]/Classify/
```

Server-side component API:

The *init* API call is used by the client-side component to initialize the server-side component by uploading the model architecture, weights, and the encryption context. From that the server can produce a homomorphic friendly version of the model to be used for inference.

The *HEcontext* API call is used by the client-side component to ask the server-side component for the homomorphic encryption parameters. Those parameters are used to generate the keys.

The *classify* API call is used by the client-side component to upload the encrypted data to the server. Once the data is received the server launch the classification. Finally, when the computations finish, the encrypted result is sent to the client-side component.

4.1.4 Privacy-preserving NN classification based on hybrid approach

4.1.4.1 Overview and main functionality

The hybrid solution uses both, the HE and 2PC, in order to build a privacy-preserving NN classification framework and maximize the efficiency of classification on deep NN. The solution is *practically* generic, namely, it supports different types of DNN (i.e., MLP, CNN, and RNN) with any number of layers, any number of neurons in each layer and any activation function (from the set of supported activation functions), while the performance still acceptable (grows linearly with the DNN's depth).
We presented a detailed design in D4.1 [1], and described how we integrated and evaluated the service in D4.2 [2].

4.1.4.2 Deployment

To deploy the service, follow the instructions in Section 3.3 (i.e. Service deployment) and Section 3.4.1 (i.e. Agent deployment).

4.1.4.3 How to use/API

In this section we describe the service API, swagger descriptions are available in Appendix 3.1.4.

Server-side component API:
There are two REST API calls supported by the server-side component: (1) *init;* and (2) *classify*. The *init* call configures the server and creates the required model. The *classify* call receives as input an encrypted vector and classifies it.

Client-side component API:
There are three REST API calls supported by the client-side component: (1) *generateKeys*; (2) *init;* and (3) *classify*. The *generateKeys* call will generate public and secret keys for homomorphic encryption. The *init* call configures the client-side component and sends configuration parameters to the server-side component. The *classify* call receives as input an encrypted vector, sends it to the server side-component for classification and receives the encrypted result. The classify API call runs interactive protocol between the client and the server components underneath. This protocol computes the layers of the NN iteratively and generically. Namely, it computes the layers of the NN one by one (depending on the layer type) on the server side and it computes the activation functions using 2PC.

4.2 Collaborative Training of Neural Network

4.2.1 Overview and main functionality

Collaborative training of NN allows multiple participants to perform a ML training collaboratively, while preserving the privacy of the training data. We presented a complete specification of the service based on Shokri and Shmatikov approach [13] in D4.1 [1], and an extension based on the method presented by Abadi et. al. [14] (presented in D3.3 [6]) in D4.2 [2]. We also described how we integrated and evaluated the service in D4.2 [2].

4.2.2 Deployment

As described in Section 3.2 and 3.3, two docker images were created, the server side and the client-side agent. Both components were implemented as Flask app with API documentation using swagger (Flask Connexion[10]). The Collaborative training supports a training of any NN with standard architecture, implemented in Keras[11].

The agent side listens on port 8080 and the server side on port 5555. The APIs documentation for both of the components can be found in the following address:

<ip>:<host>/v1/ui/

4.2.3 How to use/API

The Agent-side component exposes the following APIs (swagger images can be observed in Appendix 3.2):
get_config – allows the user to obtain the agent's training configurations
get_model – allows the client to download the trained model (either the local latest or the common collaborative)
get_model – allows the user to obtain the training status. The possible status values are:

- *WaitingToJoin* – the agent is waiting to join the training
- *WaitingForMinParticipants* – the agent joined the training; however the required minimal number of participants did not send a join request to the server
- *ReadyToTrain* – minimal number of participants asked to join/start the training and the agent can start the training
- *Training* – running the training
- *TrainingFinished* – the training successfully finished
- *TrainingTerminated* – something went wrong during the training process

init – through this call the initiator participant will initiate the training and provide necessary configurations and the model itself
join – allows the participant to join the training

[10] https://connexion.readthedocs.io/en/latest/
[11] https://keras.io/

reset – allows resetting the agent side component without restarting the docker container
set_token – when the service is integrated with IAM, through this API the client will be able to set the authentication token into the request that was sent by the agent
train- allows the user to provide the training data and start the training
Through these APIs the client app communicates with the agent-side component.

The Server-side component exposes the following APIs (swagger images can be observed in Appendix 3.2):
download_model – download the model architecture
download_weights – allows the agent to download weights of collaborative model
get_status – allows the agent to obtain the server's status:

WaitingForInitialization – the server is waiting for initialization
WaitingForMinParticipants – the server is waiting for minimal number of participants to join the training
WaitingForTraining – minimal number of participants joined the training, the server is waiting for train call
ReadyToTrain – when at least one of the participants requested to start the training, the server starts the waiting period, during that period the server allows other participants to join the training or start it
Training – training is running, new participants are not allowed to join the training or start it

init – allows the initiator (agent) to initiate the training
join – allows the agent to join the training
reset – allows the client to reset the server without restarting the docker container
train- allows the agent to request to start the training
upload_gradients – allows the agent to upload gradients calculated based on his local training
Through these APIs the agent side component communicates with the server-side component.

4.3 Clustering

4.3.1 Privacy-preserving clustering based on 2PC

4.3.1.1 Overview and main functionality

This service provides privacy-preserving trajectory clustering based on 2PC. The ABY12 library is used as the secure two-party computation (2PC) library. The main goal of this service is to provide an efficient and privacy-preserving version of the well-known Trajectory Clustering algorithm (TRACLUS) [15]. All the deployment and configuration information, and the behavioral analysis for this service were presented in the deliverable D4.2 [2]. In addition to the client-server mode, which is the general mode for our 2PC based solutions (see D3.3 [5] and D4.2 [2]), we also provide non-colluding two servers mode for this particular service. In this new mode, instead

[12] https://github.com/encryptogroup/ABY

of executing the clustering algorithm between the client-side and server-side component, the execution of the algorithm is outsourced to two non-colluding servers. Details for the new versions of API calls and service integration evaluation are presented in the deliverable D4.3 [12].

4.3.1.2 Deployment

The deployment instructions are introduced for server-side and client-side components are introduced in Section 3.3 and Section 3.4.1, respectively.

4.3.1.3 How to use/API

In this section we describe the API calls of the service. Swagger descriptions are available in Appendix 3.3.1.

Server-side component API:
There is only one REST-full API call for the server-side components in this service, namely "*start/*" API call. This API call is used for executing the clustering for both client-server and non-colluding two servers modes and this API call is automatically executed by the client-side component. Note that for all necessary communications for the execution of 2PC based clustering algorithm for both modes, the socket communication is used. Selection of modes for the clustering service is realized automatically based on the *init/* API call on the client-side component (see Client-side API below). Therefore, if *init0/* API is called in the initialization phase then, the clustering algorithm operates on the non-colluding two servers mode. Otherwise, the clustering algorithm operates on the client-server mode.

Client-side component API:
There are three API calls for the client-side component, namely *init0*, *init1* and *start*.
The *init0* API call is used for initializing the servers' information on the client side for two non-colluding servers mode. Following is an example of the *init0* API call:

curl http://[address_of_the_client]/init0/[IP_of_the_server1]/[url_of_the_server1]/[port_of_the_server1]/[IP_of_the_server2]/[url_of_the_server2]/[port_of_the_server2]/

The *init1* API call is used for the same purpose as *init0* in the client-server mode. Following is an example of the *init1* API call:

curl
http://[address_of_the_client]/init/[IP_of_the_server]/[url_of_the_server]/[port_of_the_server]/

The *start* API call is used for uploading the line segment data set as a file and starting the clustering algorithm. Following is an example of the *start* API call:

curl -F 'file=@[name_of_the_input_file]'
http://[address_of_the_client]/start/[no_of_linesegments]/[minLns]/[epsilon]/[no_of_iterations]/

4.3.2 Privacy-preserving clustering based on MinHash

4.3.2.1 Overview and main functionality

This solution provides a privacy-preserving service for the clustering of trajectories. The solution uses the MinHash algorithm, hence the name. The service uses the ABY [13] library for secure two-party computations. The solution is composed of two components, one for the client and one the server. As for the service described in Section 4.3.1, this service offers two modes. In the first one, the computations are shared between the client and the server. This performs well, but the cost of communication can be high. Thus, the second mode that rely on two non-colluding servers that will share most of the computations and communications. In the second mode the communications between the client and the server are minimized. The details of the design, deployment and integration are given in D4.1 [1], D4.2 [2] and D4.3 [12]. The details of the performance can be found in D3.3 [6].

4.3.2.2 Deployment

As described in Section 3.2 and 3.3, two docker images were created, the server side and the client-side agent.

4.3.2.3 How to use/API

In this section we describe the API calls of the service. Swagger descriptions are available in Appendix 3.3.2.

The *initv1* API call is used to initialize the client by providing the parameters to connect to the server. This call will use the client-server mode. Following is an example of the *initv1* API call:

```
curl -X POST
     -F "ipaddress=[server_ip_address]"
     -F "port=[server_port]"
     -F "url=[server_url]"
     http://[address_of_the_client]/initv1/
```

[13] https://github.com/encryptogroup/ABY

Project No. 786767

The *initv2* API call is used to provide the information related to the two servers used for the two non-colluding servers mode. Following is an example of the *initv2* API call:

```
curl -X POST
     -F "ipaddress1=[server1_ip_address]"
     -F "port1=[server1_port]"
     -F "url1=[server1_url]"
     -F "ipaddress2=[server2_ip_address]"
     -F "port2=[server2_port]"
     -F "url2=[server2_url]"
     http://[address_of_the_client]/initv2/
```

The *cluster* API call is used to upload data to the client. Following is an example of the *cluster* API call:

```
curl -X POST
     -F "trajectories=[trajectories_file]"
     -F "characteristic_trajectories=[char_trajectories_file]"
     http://[address_of_the_client]/cluster/
```

5 Platform Security and Transparency

5.1 IAM

5.1.1 Overview and main functionality

As stated in D4.3 [3], the Identity Access Manager (IAM) is in charge of providing the Authentication and Authorization services to access to the PAPAYA platform.

In order to provide Authentication and Authorization services within the Papaya platform, it is necessary to deploy the IAM server and other complementary components.

Figure 8 shows the complementary components of IAM server to provide the Authentication and Authorization services. All the accesses to the PAPAYA framework are preformed through the Security Gatekeeper components. The Security Gatekeeper component contacts the IAM to assess that the access is authenticated and authorized and then it will be redirected to the correspondent Computation Component.

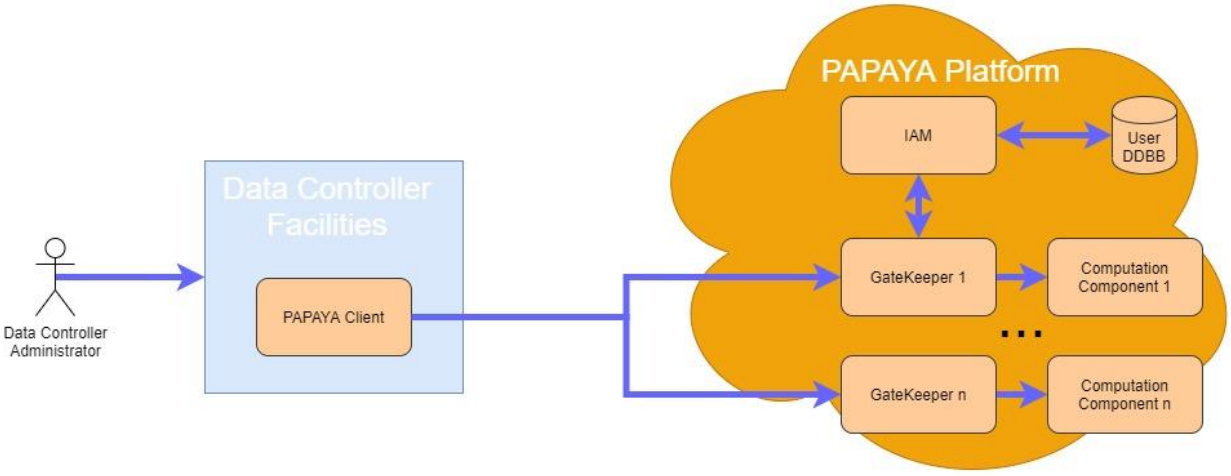


Figure 8: IAM and complementary Security Gatekeeper in PAPAYA platform [6]

5.1.2 Deployment

The deployment of IAM encompasses the following steps taken from D4.3 [12] and they are also more detailed in 0, including the commands that can be used:

1. **Set up Keycloak:** this includes:
   - secret creation needed for using them in ingress
   - application to the deployment and service

- ingress definition, that allows it will be accessible from outside and share the certificates of the initial application

if these steps are ok, then IAM is accessible

2. **Set up Gatekeeper:** in order to set up this component it is necessary to perform the following steps:
   - Generation of gatekeeper config map, configuring the mapping necessary with the specific parameters
   - Apply the Deployment + Service for the gatekeeper + example service
   - Finally, in order to allow the access to the service from outside, it is necessary to apply the corresponding ingress (see D4.3 [12]).

Testing the access: The first test must be to check the access without authentication and then the access with authentication. In the case of using authentication, the token (previously obtained) must be included in the header.

5.2 Auditing

Towards being able to hold stakeholders accountable for their use of the PAPAYA platform and services, data processing is logged both as part of the platform and locally at agents.

5.2.1 Platform auditing

As part of the Elastic stack, we deployed[14] the Filebeat[15], Logstash[16], Elasticsearch[17], and Kibana[18] components. Containers that run the analytics services log their operations to standard output, and Filebeat is configured in K8s (as part of pods) to collect all of the output and send it to Logstash.
To authenticate logs generated in the platform we created a custom Logstash filter, running as part of Logstash. The filter—named Fingerprintsign—extends the default fingerprint filter of Logstash with support for signing with ECDSA. Code is available at https://git.cs.kau.se/papaya/logstash-filter-fingerprintsign with documentation at https://git.cs.kau.se/papaya/fingerprintsign-doc . In gist, the filter has to be built, installed in Logstash, and then Logstash has to be configured to use the filter. Based on its configuration, the filter adds fields to each entry (e.g., with a signature) that is later shown in Kibana. Figure 9 summarizes the flow of logging data for platform auditing and the involved components.

[14] https://www.elastic.co/guide/en/elastic-stack/current/installing-elastic-stack.html
[15] https://www.elastic.co/beats/filebeat
[16] https://www.elastic.co/logstash
[17] https://www.elastic.co/elasticsearch/
[18] https://www.elastic.co/kibana

Project No. 786767

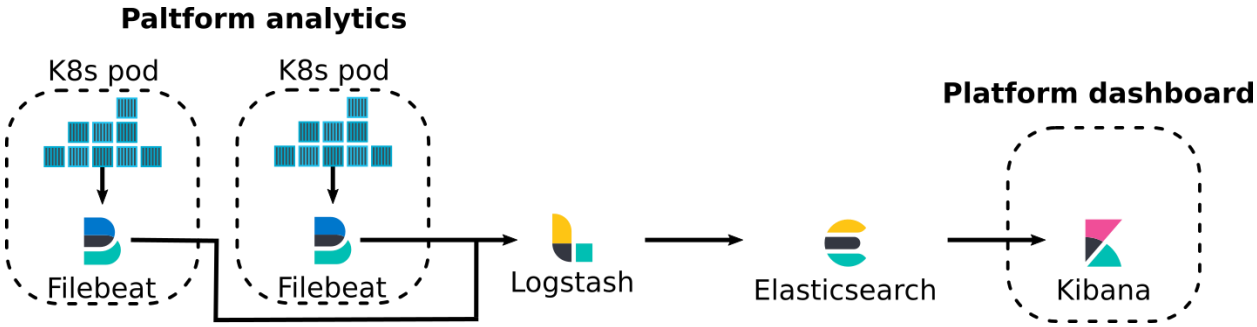


Figure 9: Platform auditing components and the flow of logs.

The Logstash component sends the signed logs to the Elasticsearch instances we run as part of the platform for storage. Users could examine the instances' (applications) logs using PAPAYA dashboard as depicted in Figure 10.

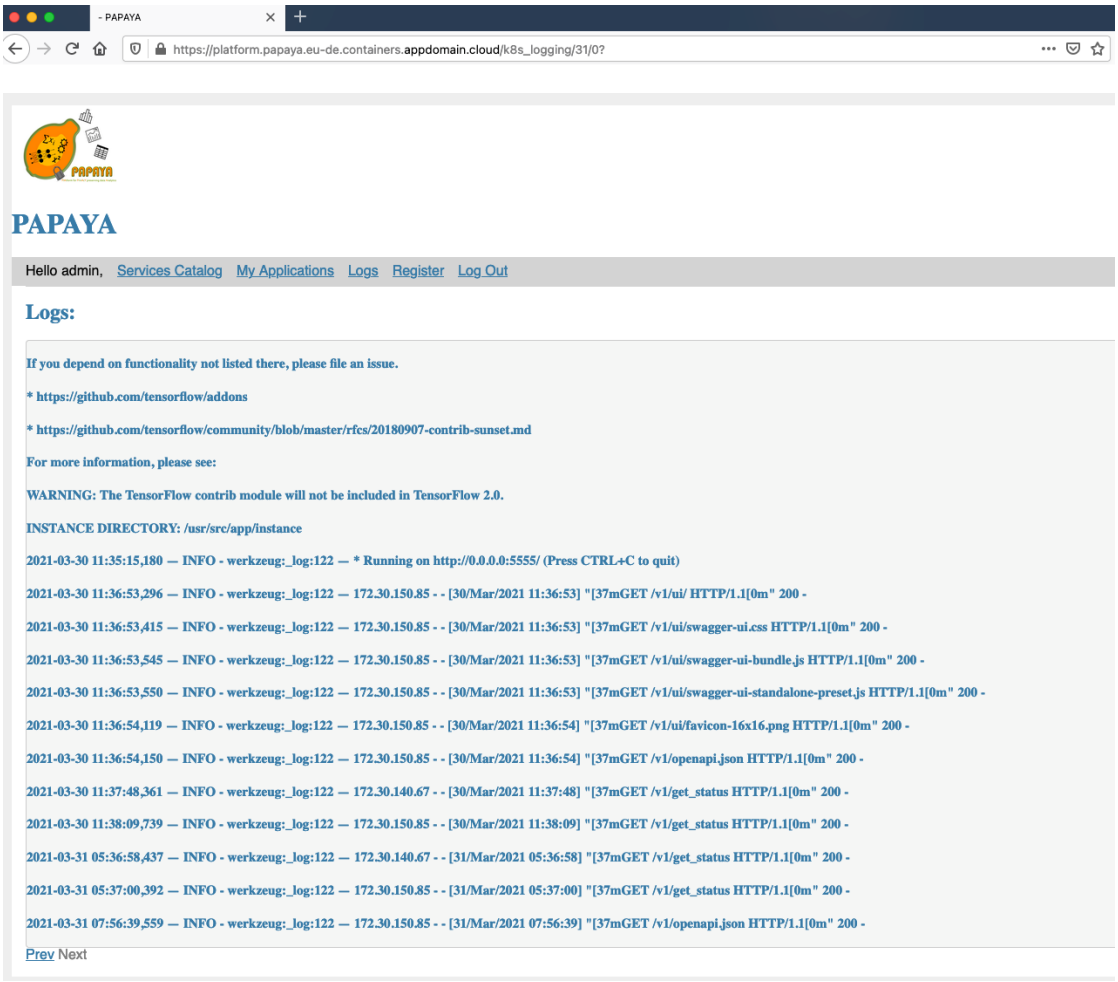


Figure 10: Instance's (server-side) operational log

5.2.2 Agent auditing

Agent Auditing is considered in the development and operational phases of a platform agent, as defined in D4.2 [2].
In the development phase, the Agent Dashboard can be used by the user of the platform agent to quickly view the generated logs generated by the agent. This is demonstrated as part of the Agent Dashboard in Section **Error! Reference source not found.**.
In the operational phase, the user of the platform agent is assumed to already be operating some logging infrastructure for its existing systems (in which the agent is integrated), as described in D4.2 [2]. Creating project-specific tooling here is therefore not wise. We note that the tooling we use for platform auditing (Elastic stack with a custom Logstash filter), as just described in Section **Error! Reference source not found.**, could be re-used by a platform client with minimal changes. The difference between the agent auditing and platform auditing settings is just the use of an appropriate method to transport logs from containers (generated by Docker or whatever container runtime that is used) into Logstash[19].

5.3 Key Manager

5.3.1 Overview and main functionality

The Key Manager (KM) component, the main aim of this component, as showed in section 4.3 of D4.2 [2], is to provide a supporting service for helping on the management of cryptographical material during the whole lifecycle of the PAPAYA project. D4.1 [1] described the architecture design, the main components of the Key Manager and the integration of the Key Manager within PAPAYA platform (in section 1.1) and more information about the configuration and deployment of this component by be found within the D4.2 [2].

5.3.2 Deployment

Key Manager server is available as a Docker image in the project's repository with the following tag: de.icr.io/papaya-de/key-manager. In order to fully deploy the Key Manager, it just necessary to execute the following command:

```
$ docker run -p 9311:9311 --name key-manager -t de.icr.io/papaya-de/key-manager
```

It is worth to highlight that, to facilitate the integration of this component with any type of implementation, the KM client is available for the following programming languages and frameworks, such as: C Sharp, Java, JavaScript, Python, etc.

5.3.3 How to use/API

In this section we describe the API calls of the Key Manager. Swagger descriptions are available in Appendix 3.4.

[19] See https://logz.io/blog/docker-logging/ for examples of how to use Filebeat or a logging driver. Accessed 2020-03-29.

Key Manager provides the following REST API calls to manage the cryptographic material:

POST/secrets - creates a Secret entity. If the payload attribute is not included in the request, then only the metadata for the secret is created, and a subsequent PUT request is required.

GET/secrets – list a project's secrets. The list of secrets can be filtered by the parameters passed in via the URL.

GET/secrets/{uuid} – retrieve a secret's metadata.

PUT/secrets/{uuid} – add the payload to an existing metadata-only secret, such as one made by sending a POST /v1/secrets.

DELETE/secrets/{uuid} – delete a secret by uuid.

GET/secrets{uuid}/payload – retrieve a secret's payload.

6 Data Subject Toolbox

Data protection by design is a key consideration of any data controller and data processor since the GDPR came into effect. The use of complex cryptographic systems—such as privacy-preserving data analytics—pose a challenge for data controllers and processors when dealing with data subjects. In particular, when it comes to the rights to data subjects to be informed about how their personal data is processed and to remain in control of this processing. Towards addressing these challenges, the PAPAYA framework includes a data subject toolbox with a number of small, independent tools that can be used by data controllers and data processors in their interaction with data subjects.

6.1 Explaining Privacy-preserving Analytics

6.1.1 Overview and main functionality

For providing enhanced transparency about the processing of personal data with PAPAYA to data subjects, the following data subject tools were developed for the data subject toolbox, which are also described in more detail in the PAPAYA deliverable D3.4 [16]

- **Tools for presenting Risk Management artefacts for assessing the impact of privacy-preserving data analytics on privacy risks**. These risk management artefacts are created by an extended version of the Privacy Impact Assessment (PIA) tool by the French Data Protection Authority CNIL as described in D3.4 [16] and include: (a) An initial threshold analysis for a PIA explaining why a PIA needed to be conducted, (b) Risk matrices that are well displayable on mobile phone screens for the privacy protection goals, where transparency, unlinkability and intervenability in addition to the classical security goals confidentiality, integrity and availability covered by CNIL's PIA tool.
- **Tools for explaining and exemplifying how privacy-preserving data analytics work**. A focus was on tools for explaining with multiple layers of details privacy-preserving neural networks for classification and collaborative training. To this end, the tools were developed to explain privacy-preserving building blocks of homomorphic encryption, multi-party computation (MPC) and two-party computation (2PC), functional encryption as well as differential privacy for collaborative learning. Please refer to PAPAYA D3.4 [16] for further details on the user interface design development.

6.1.2 Deployment

While our tools for explaining the functionality and impact of privacy-preserving data analytics can enhance both ex ante and ex post transparency[20], we regard them as especially useful for

[20] The concept of transparency comprises both 'ex ante transparency', which enables the anticipation of consequences before data are actually disclosed (e.g., with the help of privacy policy statements), as well as 'ex post transparency', which informs about consequences if data already have been revealed (e.g.,

enhancing ex ante transparency when data subjects are requested to give consent to the processing of their data. For this, the tools can be integrated in multi-layered policy notices, following a recommendation by the Art. 29 Data Protection Working Party for enhancing the usability of privacy notices [17]. Information about the threshold analysis of the conducted PIA, residual privacy risks and explanations of what type of protection can be achieved by privacy-preserving data analytics and how it works can be provided on lower layers.

To this end, a clickable link for "More on <The organisation's> Privacy Impact Assessment and Privacy by Design approach" could be added on the top layer of a privacy notice. Figure 11 shows such multi-layered policy notices can look like, which links to the data subject tool for explaining functional encryption and links to the risk management artefacts that were obtained by conducting a PIA for the specific use case.

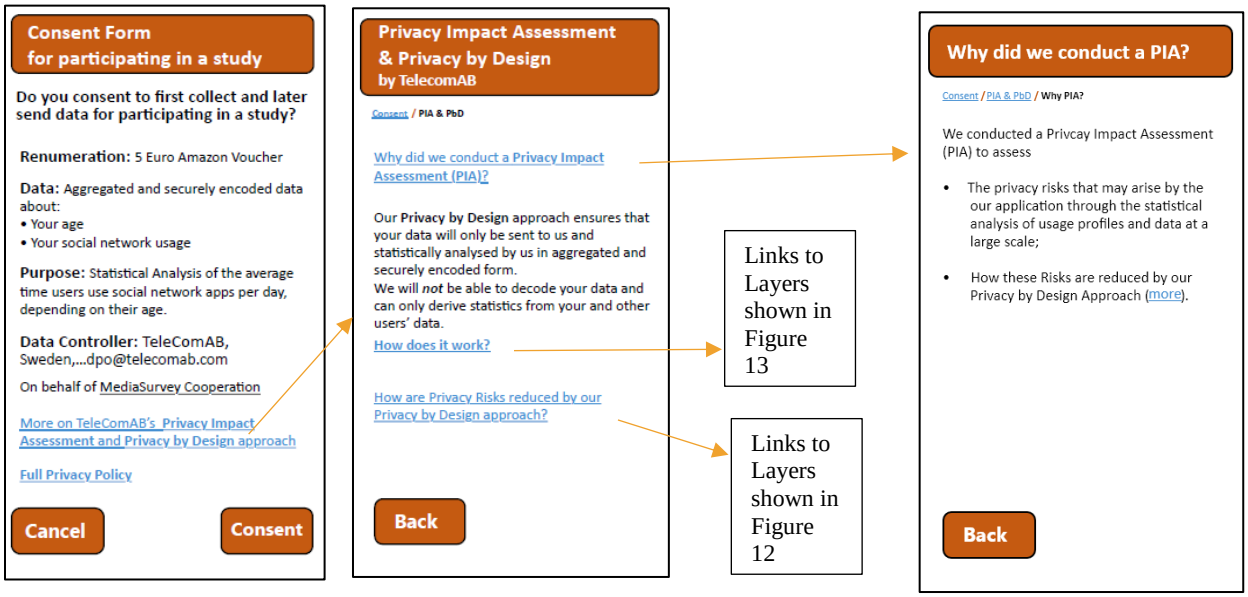


Figure 11: Top layer (on the left), secondary layer in the (middle) and third layers of a multi-layered Privacy Notice [17]. The Second Layer links to information about the initial threshold analysis that can be displayed on a third layer and to further layers displayed in Figure 13 and Figure 12.

what data are processed by whom and whether the data processing is in conformance with negotiated or stated policies) **Invalid source specified.**.

Project No. 786767

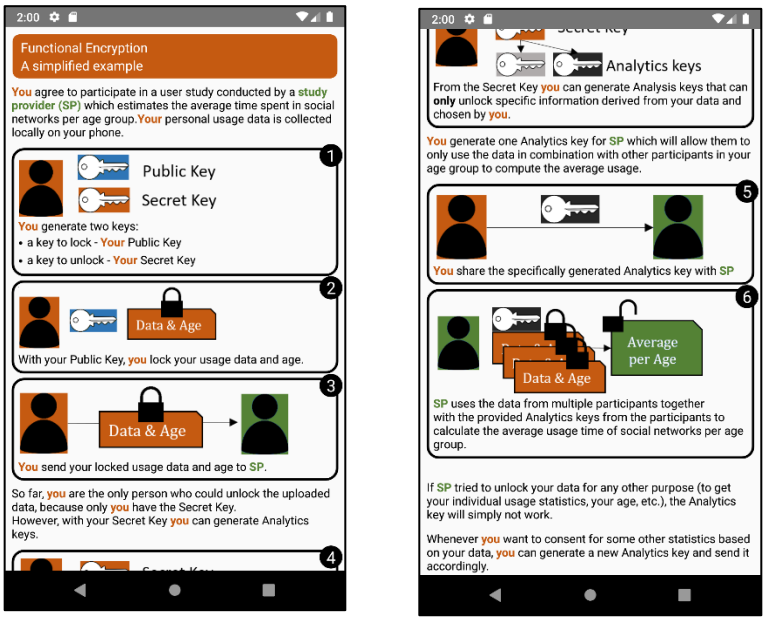


Figure 13: Data subject tool with scrollable user interface explaining how functional encryption works.

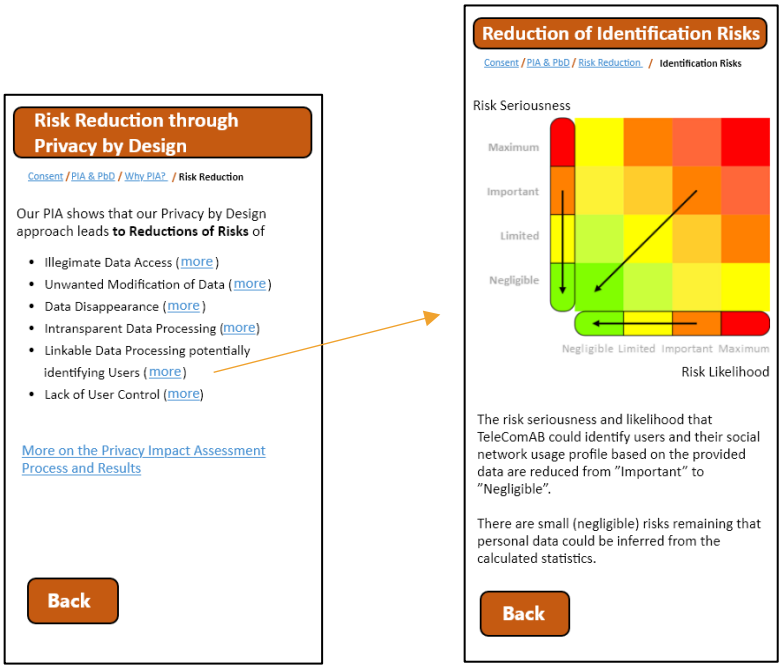


Figure 12: User interfaces (appearing on a third and fourth layer of a multi-layered privacy notice) showing how PAPAYA is impacting/reducing the risk seriousness and risk likelihood of linkability (and thus of identification).

6.1.3 How to use/API

The data subject tools for explaining privacy-preserving analytics are React Native components, easily integrated into existing mobile apps or websites. They can be accessed via the following repositories (including source code, documentation, and screenshots and/or demos):

- Data subject tool for explaining 2PC: https://git.cs.kau.se/papaya/gatsby-papaya-encryption-data-analysis
- Data subject tool for explain MPC: https://git.cs.kau.se/papaya/react-native-papaya-mpc
- Data subject tool for explain differential privacy for collaborative learning: https://git.cs.kau.se/papaya/react-native-papaya-differential-privacy
- Data subject tool for explaining Functional Encryption: https://git.cs.kau.se/papaya/react-native-papaya-functional-encryption

For producing the risk management artefacts, a PIA needs to be conducted with the extended PIA tool, which is available as AppImage on Linux, and Installation on Windows at https://hex.cse.kau.se/~jonamagn/ . Source code is also available at https://github.com/papaya-h2020/pia.

The tool produces a JSON-file and visual elements (.png) that present a risk overview with one risk matrix in regard to all six privacy protection goals mentioned above, risk mapping and the proposed action plan. The JSON -file can then be used to produce the six independent risk matrices plus the initial threshold analysis for the display on mobile devices.

For this, the following instructions have to be followed:

- Clone the git repository https://git.cs.kau.se/papaya/react-native-papaya-riskmatrix to your local machine and initialize the project by running ‘npm install’ which installs all dependencies for the tool.
- Follow the example located in the ‘examples/’ folder that shows how the matrices are produced.
- Place the JSON -file in an accessible location within the project structure or replace the existing json-file. Build an installable android application by running the command ‘npm run android:buildDev’.

Integrate this in to an existing application by following the official React Native guide at https://reactnative.dev/docs/integration-with-existing-apps .

6.2 Data Disclosure Visualization Tool

6.2.1 Overview and main functionality

The Data Discloser Visualization Tool provide transparency to data subjects for an application about what actors process or have access to what types of their personal data.

The tool provides the data subject with three different views that can be activated via three tabs on the top of the user interface: The trace view, the actor view, and the data view. The trace view is building om our earlier research on usable transparency [18], while the alternative actor and data views are motivated by [19].

- **The trace view** screens (see Figure 14) are divided into 2 parts: The “Data” upper half of the screen shows all types of personal data (with an icon and a name) of a data subject that will be processed for this application. The “Actor” lower half of the screen shows all actors including the data subject (also with an icon and name), who process or have access to data of the data subject. If the user clicks on an actor icon, traces appear to the data that the actor can access/process. The user can also click on a data icon, in which case traces appear showing the actors who process or have access to this type of data. Conversely, if the user clicks on an actor icon, traces to the icons of data appear to that this actor processes or has access to.

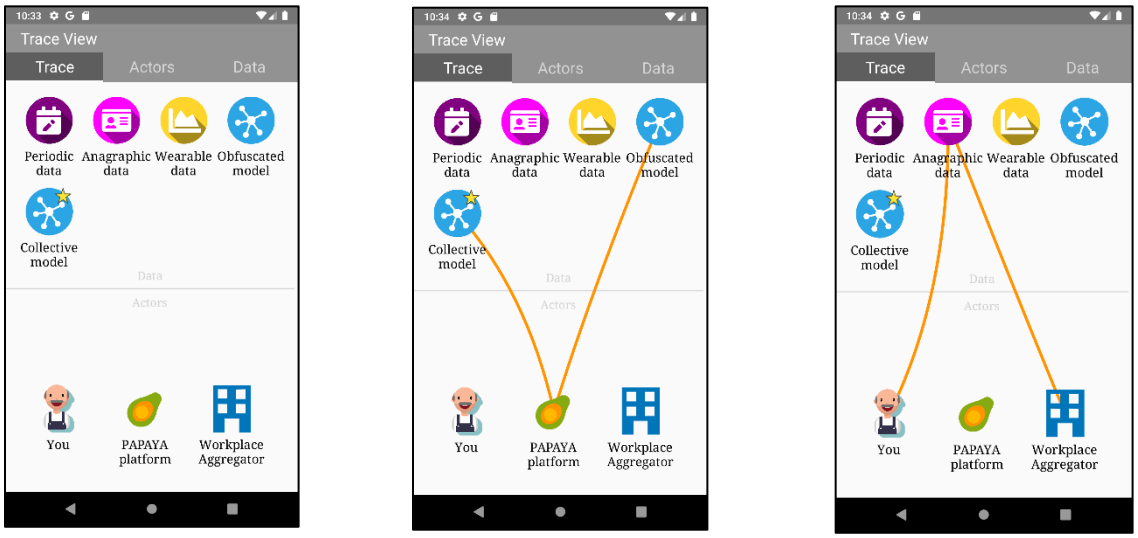


Figure 14: Trace view user interfaces (illustrated for the PAPAYA use case 2 application).

- **The actor view** (see Figure 15) provides a description for all actors for the application who have access to and/or process the user’s personal data. Moreover, in complementation to the trace view, it provides an alternative way of getting an overview of the type of data that an actor processes or has access to.

Project No. 786767

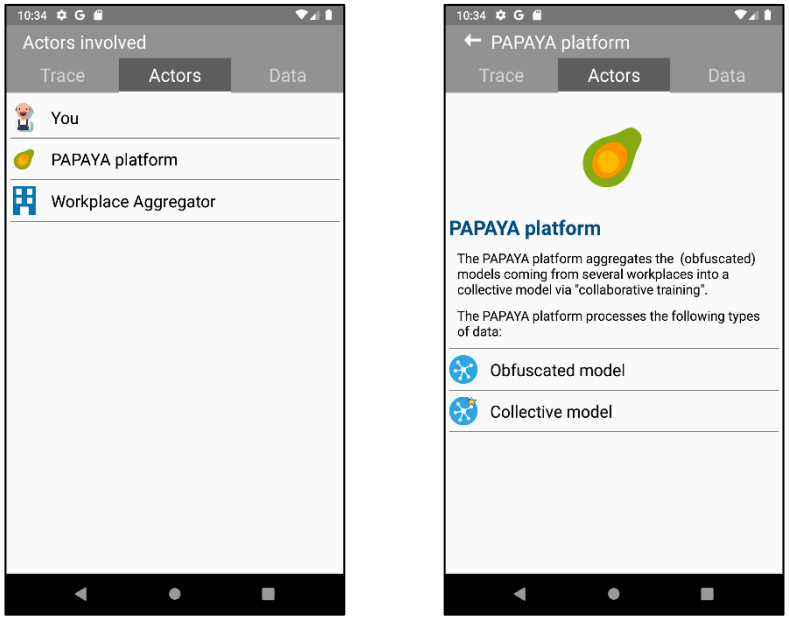


Figure 15: Actor-view, which provides descriptions of the actors and what types of data they process or can access

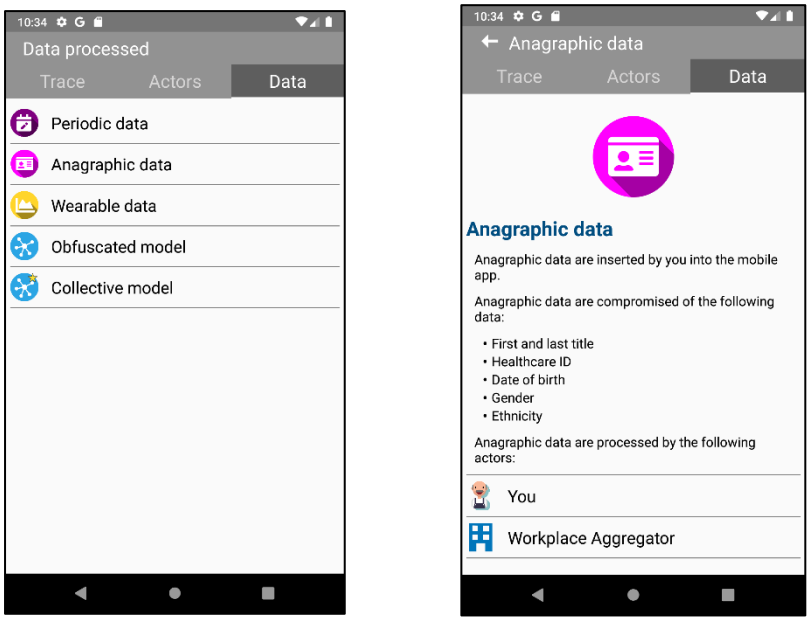


Figure 16: Data view, which provides descriptions of the data types that lists the actors that can access or process the data.

- **The data view** (see Figure 16) provides a description for all types of personal data that can be accessed within an application. In complementation to the trace view, it also provides an alternative way of getting an overview of actors has access to these types of data.

6.2.2 Deployment

The Data Discloser Visualization Tool can be used both for providing ex ante or ex post transparency. For enhancing ex ante transparency, the tool could be activated via a link in a multi-layered privacy notice, similarly to the integration of the tools for explaining privacy-preserving machine learning as shown in section 6.1.2.

6.2.3 How to use/API

For installation and integration of the tool, the user should follow these instructions:

- Clone the git repository located at: https://git.cs.kau.se/papaya/react-native-papaya-trace-viewer
- Make sure that 'npm' installed and run 'npm install' in the main repository folder to install all dependencies needed to run the tool.
- In order to change the content of the Data Discloser Visualization Tool one edits the file located in 'src/' named 'TraceData.js' that contains a javascript object that controls text data and data related to how data subjects and actors are connected.
- To produce an installable android application run the command 'npm run android:buildDev' in the root folder and the path to the .apk should be returned. To integrate the tool into and existing application follow the official guide for react native at https://reactnative.dev/docs/integration-with-existing-apps
- To produce an web version of the tool run the command 'npm run web:build' in the root folder and all web content should be available in the 'dist/' folder.

Look into the README of the project for more detailed information.

6.3 Annotated Log View Tool

6.3.1 Overview and main functionality

The Annotated Log View tool provides a timeline of log events, each event representing either a generic event or a PAPAYA event. The generic events are intended to be tailored by the user of the tool to explain data processing related to their use of personal data. The special PAPAYA Event relates to one of the data analytics developed in the project and that has a tool developed for explaining it (see Section 6.1). The PAPAYA event then launches the view for explaining the used data analytics, see Figure 17 for an example.

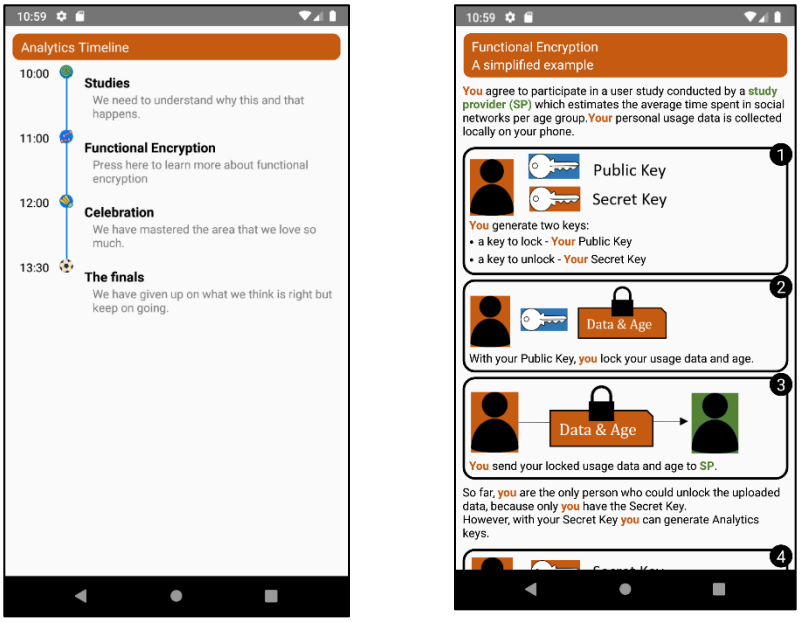


*Figure 17: Generic timeline view on the left, where clicking on events related to analytics from PAPAYA expands to the Explaining Privacy-preserving Analytics views (see Section **Error! Reference source not found.**), as shown to the right for Functional Encryption.*

6.3.2 Deployment

The purpose of the Automate Log View tool is to provide ex post transparency and explain personal data processing to users. The tool could be integrated as part of transparency or privacy dashboards within apps, offering a personalized view of the processing of their personal data in chronological order.

6.3.3 How to use/API

For installation and integration of the tool, the user should follow these instructions:

- Clone the git repository located at: https://git.cs.kau.se/papaya/react-native-papaya-timeline
- Make sure that 'npm' installed and run 'npm install' in the main repository folder to install all dependencies needed to run the tool.
- In order to change the content of the Annotated Log View Tool one edits the file located in 'src/' named 'TimelineData.js' that contains a javascript object that is a representation of the view.
- To produce an web version of the tool run the command 'npm run web:build' in the root folder and all web content should be available in the 'dist/' folder.

Look into the README of the project for more detailed information.

6.4 Privacy Engine

6.4.1 Overview and main functionality

The Privacy Engine (PE) provides two main functionalities (as stated in D3.2 [17]):

- Privacy Preferences Manager (PPM): provides all the functionality to the data subjects to define the privacy preferences over their personal data then this data can be collected for processing by big data analytics tasks in a privacy-preserving manner. The way that the Privacy Engine performs this task is by transforming high-level descriptions to machine-readable policies. Once the user sets their preferences, it will only be permitted to process the data allowed by the data subject, for example it can be filtered and excluded certain personal attributes [17].

- Data Subject Rights Manager (DSRM): PE allows the data subject to exercise their rights according with GDPR (e.g. the right to erasure his/her personal data). This is performed firstly by allowing the data controller to select the communication channel to execute the right that the subject will wish to exercise (email, publisher/subscriber pattern, protection orchestrator) and finally it provides a user centric GUI to the data subject to exercise his/her rights [17].

6.4.2 Deployment

As described in Section 6.4 of D4.2 [2] and D4.3 [12], the deployment of the PE components is as follows:

- The **back-end components** can be deployed using Docker containers. There are two images correspondent to the mentioned functionalities of the Privacy Engine which are PPM and DSRM. The command to deploy PPM is as follows:

```
$ docker run -p 8080:8080 de.icr.io/papaya-de/privacy_engine-ppm-server:latest
```

And the command to deploy DSRM container:

```
$ docker run -p 8080:8080 de.icr.io/papaya-de/privacy_engine-dsrm-server:latest
```

- The **Front-End** interfaces have been developed using the NativeScript framework and it have been created a mobile APK which can be deployed in a mobile device. In order to integrate the interfaces with the final pilots applications, it can be follow this guide: https://www.nativescript.org/faq/how-do-i-add-nativescript-to-an-existing-ios-or-android-app.

6.4.3 How to use/API

The Privacy Engine provides a set of services that can be classified by considering its two main functionalities that are the Privacy Preferences Manager (PPM) and the Data Subject Rights Manager (DSRM).

Privacy Preferences Manager (PPM)
The Privacy Preferences Manager (PPM) allows to the data subject to decide and set his/her privacy preferences regarding his/her collection and use of their personal data. To perform this task, there is a web interface that prepares and uses services (this will be done by the Privacy Expert) to store customized questions that will be asked to the data subject.

The first step is in the Figure 18. In this screen the privacy expert (PE) is preparing a question that will be answered by the end user. He/she must fill in the title of the question and the content that will be asked to the user and the server in which it will be stored the question. At the end the privacy expert clicks on “Next” button.

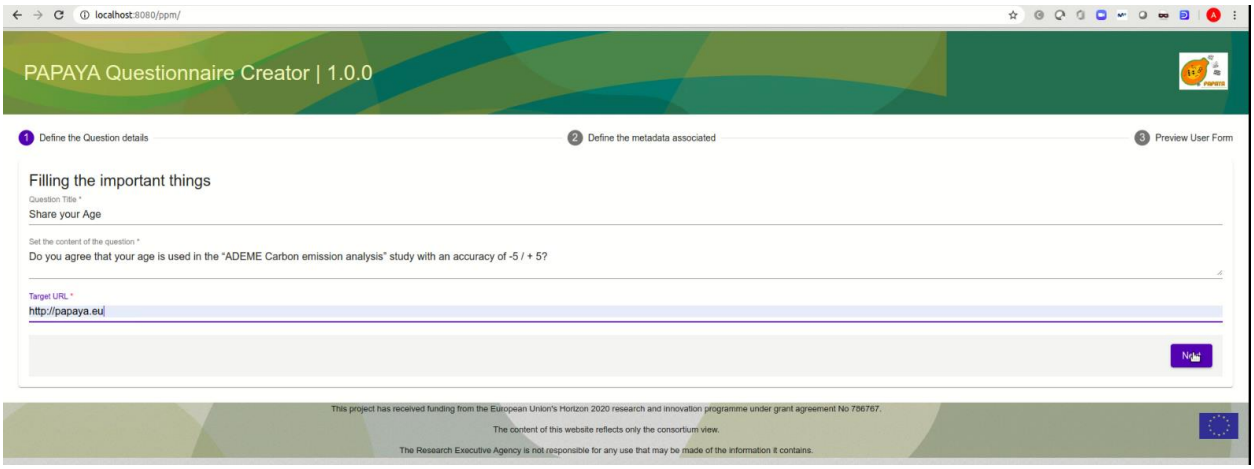


Figure 18: PPM: creating a question (first step)

In the second step there will be defined the metadata for the question, and it is shown in Figure 19. In this case there is a screen divided in two frames, the left one is the form that the PE must fill in and the right one is the definitions of all the selections that he /she can use to fill in the form. There are several definitions: type of data, processing that will be performed, purpose of collecting the data, who will receive the data, the location and duration of the data storage, all the possible values of them are included in the definitions frame.

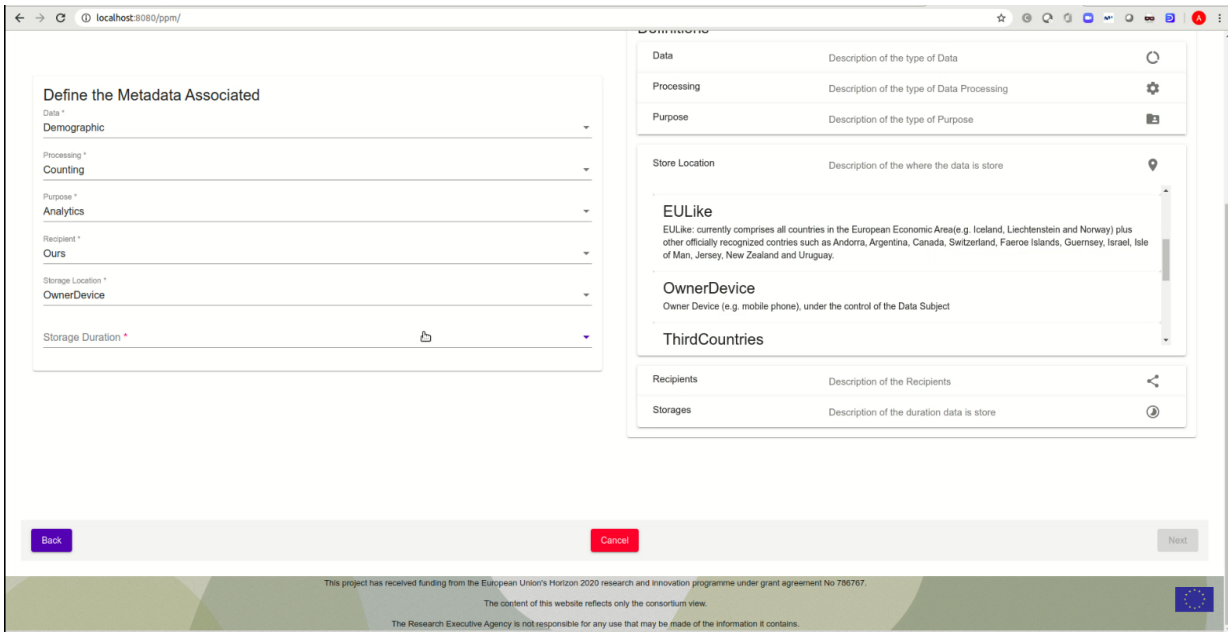


Figure 19: PPM: second step, metadata definition for the question.

After that, the PE have a preview of the question that he/she has created (see Figure 20). On the left frame there is the HTML definition, and on the right frame there is the question as will be presented to the user. And if everything is correct, he/she can submit the question and continue with another question if needed.

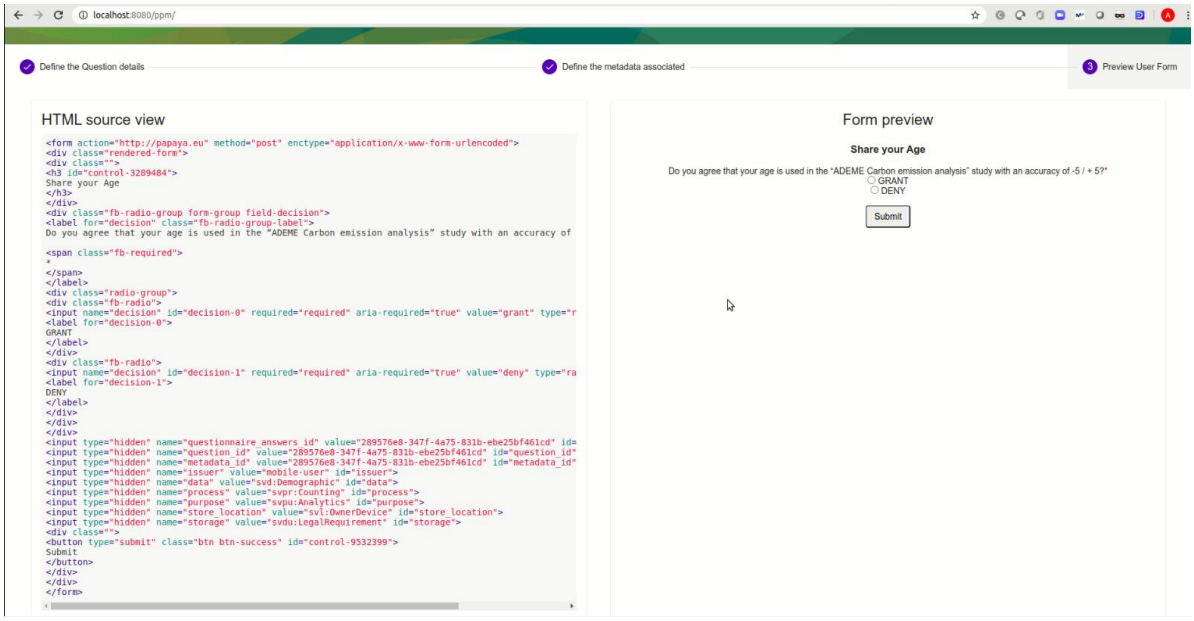


Figure 20: PPM: question preview, in HTML and real format

Data Subject Rights Manager (DSRM)
The Privacy Engine through DSRM allows to the data subject to exercise his/her rights, every right will be exercised in the way selected by the Privacy Expert. To this aim, the DSRM allows the use of the services that will configure the way to exercise every right by using a web interface. As a first step, the Privacy expert accesses to the interface of The Data Subject Right manager. In this interface he/she can find all the rights (right to be informed, right of access, right to rectification, right to erasure, right to restrict processing, right to data portability and right to object). For each right he/she must fill in the form and can choose between three ways of performing the action: by using email configuration, by using publish subscribe methods or using the protection orchestrator.

In the case of using email configuration (see Figure 21), in which the privacy expert fills in the email to which it must be sent (this person knows the actions that must be followed to execute this right), the subject of the email and the description. At the end, the privacy expert will click on the button send (there are two, in the case that he/she is only configuring this right it can be used the first one, but in case he/she has configured several rights it can be used the button “Send all configurations”).

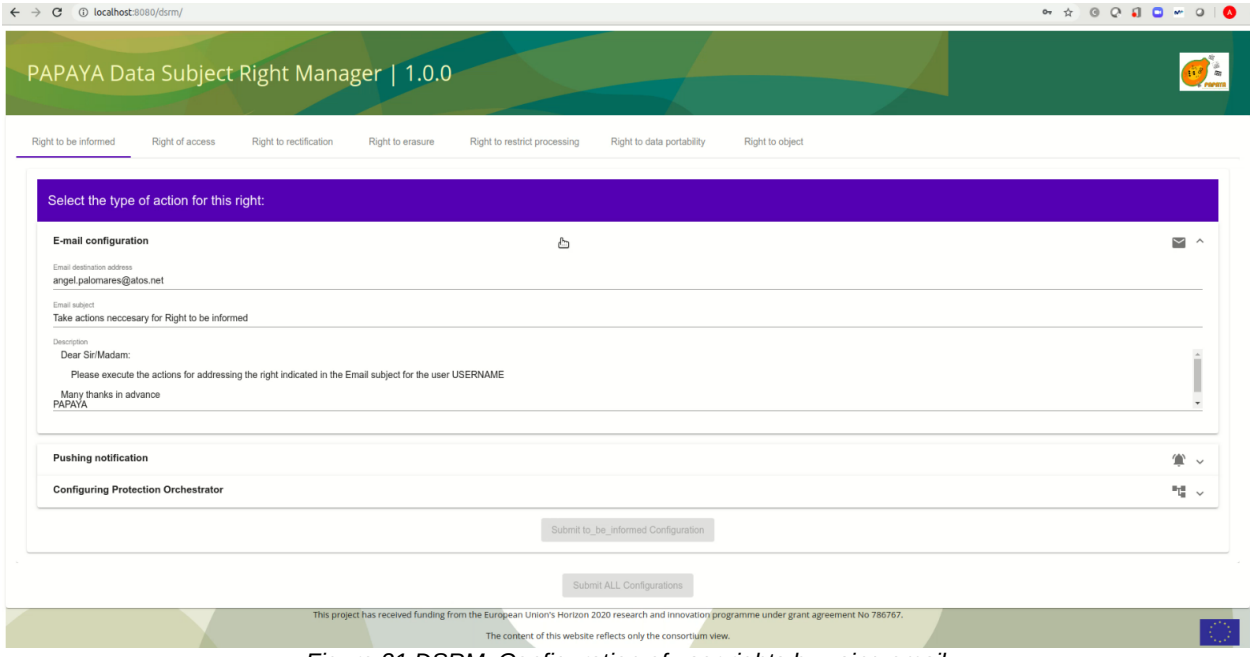


Figure 21:DSRM: Configuration of user rights by using email

In the case of using publish/subscribe methods, in this case, the needed data are the server that will listen to this and the right that is being configured. See Figure 22 in which it is being configured the right of access using this method.

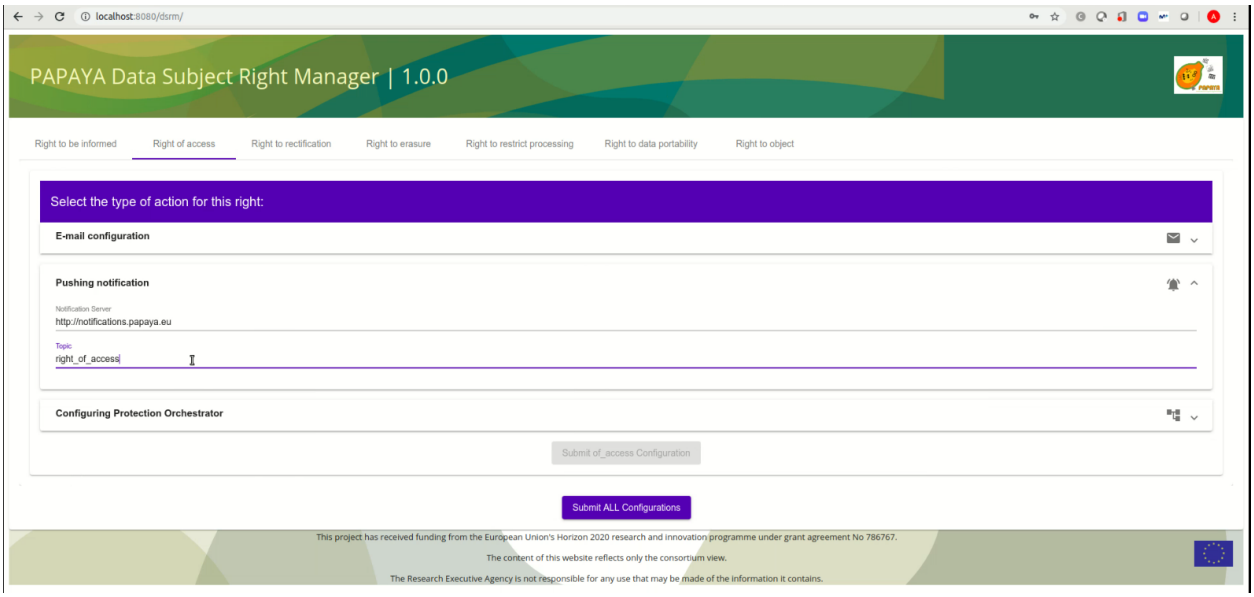


Figure 22: DSRM: Configuration of user rights by using publish subscribe methods

7 Conclusions

The main goal of the PAPAYA platform is to be used by service developers to deploy and run privacy-preserving services, and to be used by service consumers to employ privacy-preserving analytics. We presented platform functional design and architecture in the deliverables D4.1 [1], D4.2 [2], and D4.3 [3]. We explained how different services can be integrated into the platform in a way that they will be interoperable/compatible with each other and could work together in the integrated platform. In particular, by using IBM Kubernetes cloud service, we show that the analytics are possible to run using modern cloud environments; by adding Identity and Access management (IAM), we ensure that access to the analytics can be properly authenticated and authorized; by adding auditing mechanisms, we ensure that the analytics generate appropriate logs that can be centrally collected. We also presented the design of platform dashboards that provides UI, configuration, and visualization functionality, and described how we deployed all the services on the IBM Kubernetes cloud account and how we evaluated their integration.

In this deliverable, which represents one of the outputs for Task T5.3 (i.e., “Technology assessment and recommendations”), we summarized all functionality and workflows of the PAPAYA platform and provided a detailed description of how the users should interact with the system, including deployment instructions and API for each platform service and/or tool. This document could help service developers to understand how to deploy new services on the PAPAYA platform, and could help platform/service users to figure out what services and tools are available on the PAPAYA platform, and how to incorporate them in their applications.

References

[1] B. Rozenberg, R. Shmelkin, B. Bozdemir, O. Ermis, M. Önen, M. Barham, M. Azraoui, S. Canard, B. Vialla and T. Pulls, "D4.1-Functional Design and Platform Architecture".

[2] B. Rozenberg, R. Shmelkin, B. Bozdemir, O. Ermis, M. Önen, M. Barham, M. Azraoui, S. Canard, B. Vialla and T. Pulls, *D4.2 - Progress report on platform implementation and PETs integration,* 2020.

[3] B. Rozenberg, R. Shmelkin, B. Bozdemir, O. Ermis, M. Önen, S. Canard, B. Vialla and T. Pulls, "D4.3 Final report on platform implementation and PETs integration," 2021.

[4] B. Bozdemir, O. Ermis, M. Önen, M. Barham, M. Azraoui, S. Canard, B. Vialla, B. Rozenberg and R. Shmelkin, *D3.1 - Preliminary Design of Privacy Preserving Data Analytics,* 2019.

[5] S. Canard, B. Vialla, B. Bozdemir, O. Ermis, M. Önen, M. Barham, B. Rozenberg, R. Shmelkin, I. Adir and R. Masalha, *D3.3 - Complete Specification and Implementation of Privacy preserving Data Analytics,* 2020.

[6] S. Canard, B. Vialla, B. Bozdemir, O. Ermis, M. Önen, M. Barham, M. Azraoui, B. Rozenberg and R. Shmelkin, *D3.3 - Complete Specification and Implementation of Privacy preserving Data Analytics,* 2020.

[7] B. Zvika, G. Craig and V. Vinod, "Fully Homomorphic Encryption without Bootstrapping".

[8] Z. Brakerski, "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP," in *Annual Cryptology Conference*, 2012.

[9] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," *Cryptology ePrint Archive,* vol. Report 2012/144, 2012.

[10] J. H. Cheon, A. Kim, M. Kim and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*, Hong Kong, China, 2017.

[11] Microsoft Research, Redmond, WA., *Microsoft SEAL,* : https://github.com/Microsoft/SEAL, 2018.

[12] B. Rozenberg, R. Shmelkin, B. Bozdemir, O. Ermis, M. Önen, S. Canard, B. Vialla and T. Pulls, "D4.3 Final report on platform implementation and PETs integration," To be submitted in 2021.

[13] R. Shokri and V. Shmatikiv, "Privacy-preserving deep learning," in *53rd Annual Allerton Conference on Communication, Control, and Computing*, Allerton, 2015.

[14] M. Abadi, A. Chu, I. Goodfellow, B. H. McMahan, I. Mironov, K. Talwar and L. Zhang, "Deep learning with differential privacy," in *the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[15] J.-G. Lee, J. Han and K.-Y. Whang, "Trajectory Clustering: A Partition-and-Group Framework," in *SIGMOD*, 2007.

[16] S. Fischer-Hübner, M. T. Beckerle, J. S.¨ Pettersson and P. Murmann, "D3.4 - Transparent Privacy preserving Data Analytics," PAPAYA report document, 2020.

[17] Art 29 Data Protection Working Group, "Guidelines on transparency under Regulation 2016/679," 2016.

[18] J. Angulo, S. Fischer-Hübner, T. Pulls and E. Wästlund, "Usable transparency with the data track: a tool for visualizing data disclosures.," in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, 2015.

[19] D. Wilkinson, "Privacy at a glance: the user-centric design of glanceable data exposure visualizations.," *PoPETS*, pp. 416-435, 2020.

Appendix 1 Platform Dashboard Deployment files

deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: platform-dashboard
  name: platform-dashboard
spec:
  replicas: 1
  selector:
    matchLabels:
      app: platform-dashboard
  strategy:
    rollingUpdate:
      maxSurge: 1
```

```
    maxUnavailable: 1
  type: RollingUpdate
revisionHistoryLimit: 5
template:
  metadata:
    labels:
      app: platform-dashboard
  spec:
    containers:
    - name: platform-dashboard
      env:
      - name: LC_ALL
        value: C.UTF-8
      - name: FLASK_APP
        value: papaya_server
      - name: LANG
        value: C.UTF-8
      - name: ADMIN_USERNAME
        valueFrom:
          secretKeyRef:
            name: platform-admin
            key: username
      - name: ADMIN_PASSWORD
        valueFrom:
          secretKeyRef:
            name: platform-admin
            key: password
      image: de.icr.io/papaya-de/platform_dashboard
      imagePullPolicy: Always
      volumeMounts:
      - mountPath: /papaya_server/instance/
        name: platform-dashboard-vm
      ports:
      - containerPort: 5000
        protocol: TCP
    serviceAccountName: papaya-sa
    volumes:
    - name: platform-dashboard-vm
      persistentVolumeClaim:
        claimName: platform-dashboard-pvc
```

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: platform-dashboard
  name: platform-dashboard
```

```
spec:
  ports:
  - name: http
    port: 5000
    protocol: TCP
    targetPort: 5000
  selector:
    app: platform-dashboard
```

ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    ingress.bluemix.net/redirect-to-https: "true"
  name: platform-dashboard
  namespace: papaya
spec:
  rules:
  - host: papaya.eu-de.containers.appdomain.cloud
    http:
      paths:
      - backend:
          serviceName: platform-dashboard
          servicePort: 5000
        path: /
  tls:
  - hosts:
    - papaya.eu-de.containers.appdomain.cloud
    secretName: papaya
```

pvc.yaml

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: platform-dashboard-pvc
  labels:
    type: ibmc-file-gold
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 20Gi
  storageClassName: ibmc-file-gold
```

Appendix 2 IAM installation

Main steps taken from this link: https://www.openshift.com/blog/adding-authentication-to-your-kubernetes-web-applications-with-keycloak

These are the steps:

1. Set up Keycloak
   - To set up the secret that we will use later on for ingress (with the certificates). Note that this step is not necessary for real deployment

     ```
     kubectl apply -f papaya.yaml
     ```
   - To apply Deployment+Service

     ```
     kubectl apply -f keycloak.yaml
     ```
   - define the ingress in order it will be accesible from outside

     ```
     kubectl apply -f keycloak-ingress.yaml
     ```

   If everything has working properly then we can access to the IAM using https//iam.papaya.eu-de.containers.appdomain.cloud
   In the case using minikube, first obtain the IP used by minikube and then override /etc/hosts with the new IP and domain specified.

2. Setup Gatekeeper

   To setup the Gatekeeper we followed the following article: (https://dev.to/techworld_with_nana/how-to-setup-a-keycloak-gatekeeper-to-secure-the-services-in-your-kubernetes-cluster-5d2d)

   ```
   kubectl create secret generic gatekeeper --from-file= ./gatekeeper.yaml -n my-namespace
   ```

   - generate the gatekeeper config map. The configuration of the config map will contain the main configuration parameters used by the gatekeeper:

     ```
     kubectl apply -f gatekeeper-configmap.yaml
     ```
   - Apply the Deployment + Service for the gatekeeper+example service

     ```
     kubectl apply -f sp-example.yaml
     ```
   - Finally, in order to access to the service from outside, apply the corresponding ingress

     ```
     kubectl apply -f sp-example-ingress.yaml
     ```

3. Testing the Access:

- Accessing without authentication:

```
curl -X GET https://proxy.papaya.eu-de.containers.appdomain.cloud/user
```

- Accessing with authentication:
  - Obtain access token

```
curl -X POST "https://iam.papaya.eu-
de.containers.appdomain.cloud/auth/realms/papaya-
realm/protocol/openid-connect/token" \
-H "content-type: application/x-www-form-urlencoded" \
-d "client_id=papaya-proxy-client" \
-d "client_secret=3712874a-4555-4062-b9d0-4750c5c1a16a"  \
-d "username=papaya" \
-d "password=papaya"  \
-d 'grant_type=password'
```

Or you can use the following script to obtain the token:

```
./getAccessToken.sh
Using Keycloak: https://iam.papaya.eu-
de.containers.appdomain.cloud/auth/realms/papaya-
realm/protocol/openid-connect/token
realm: papaya-realm
client-id: papaya-proxy-client
username: papaya
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2346  100  2214  100   132   6668    397 --:--:-- --:--:-- --:--:--  7066
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiS...
```

  - Include in the header

```
curl -X GET https://proxy.papaya.eu-de.containers.appdomain.cloud/admin
-H "Authorization: Bearer eyJhbG..."
```

Project No. 786767

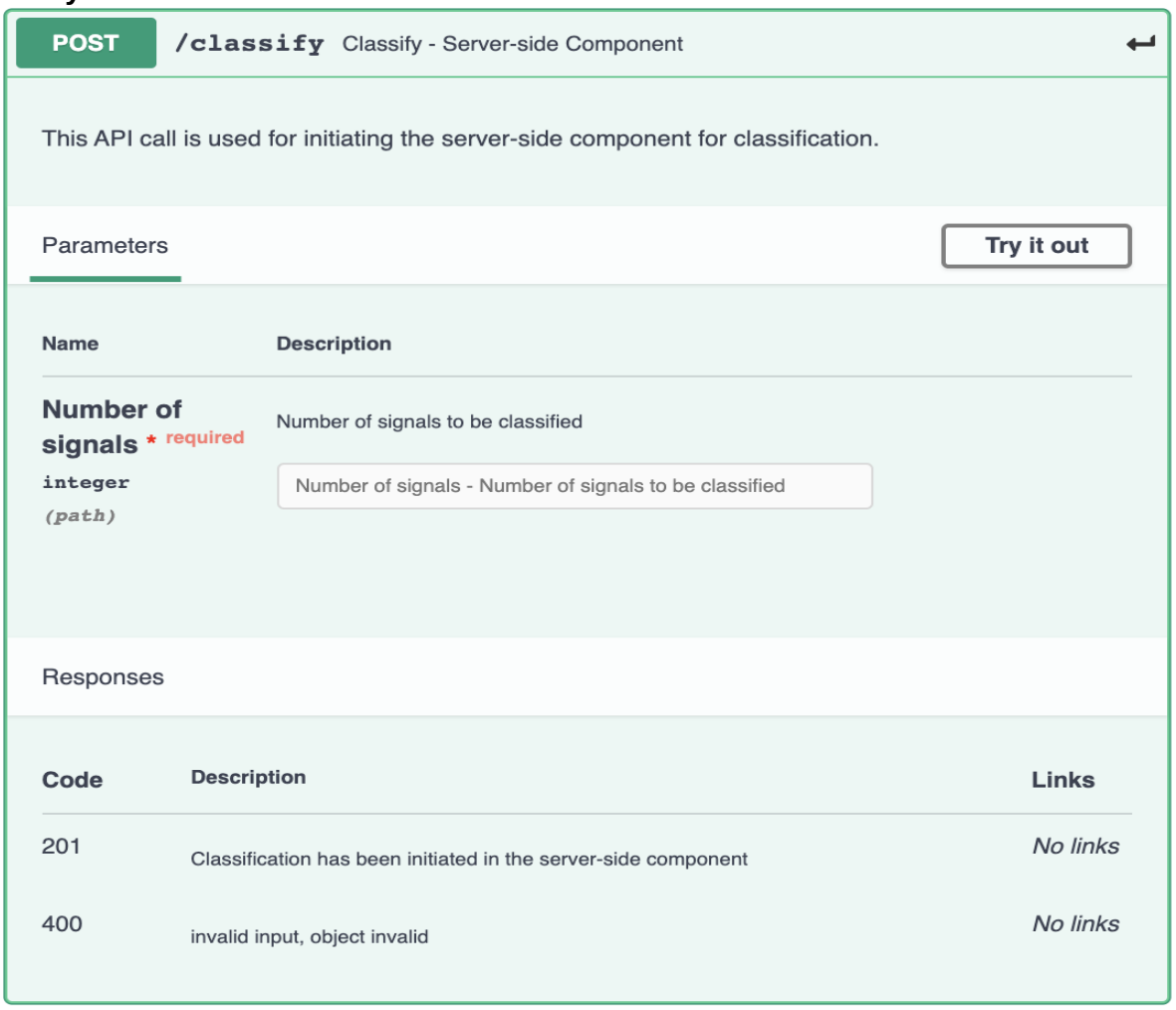
Appendix 3 PAPAYA Services/Tools - API

1. Apply Neural Network Model

1.1. Privacy-preserving NN classification based on 2PC

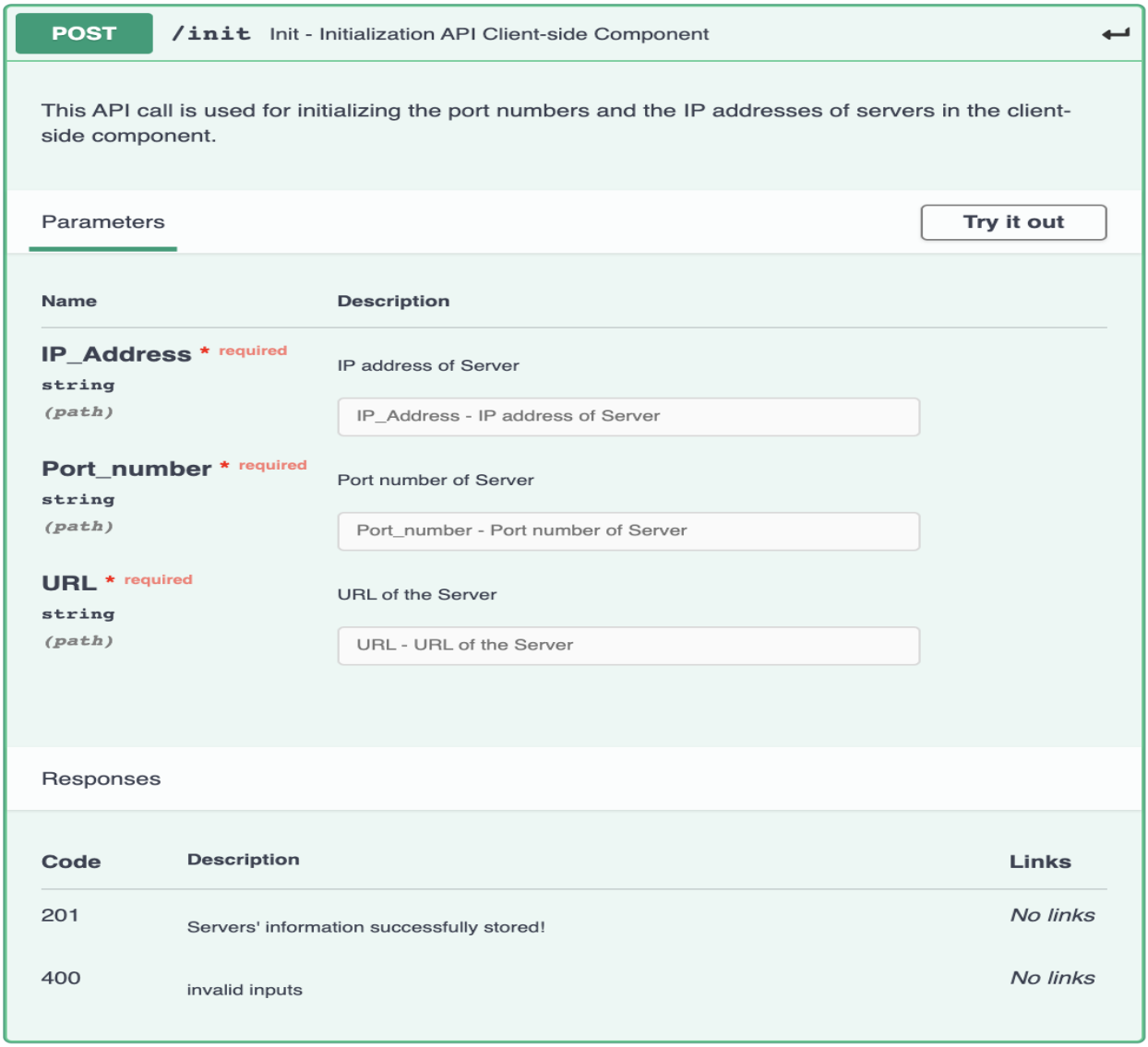
Server-side component API:

Classify



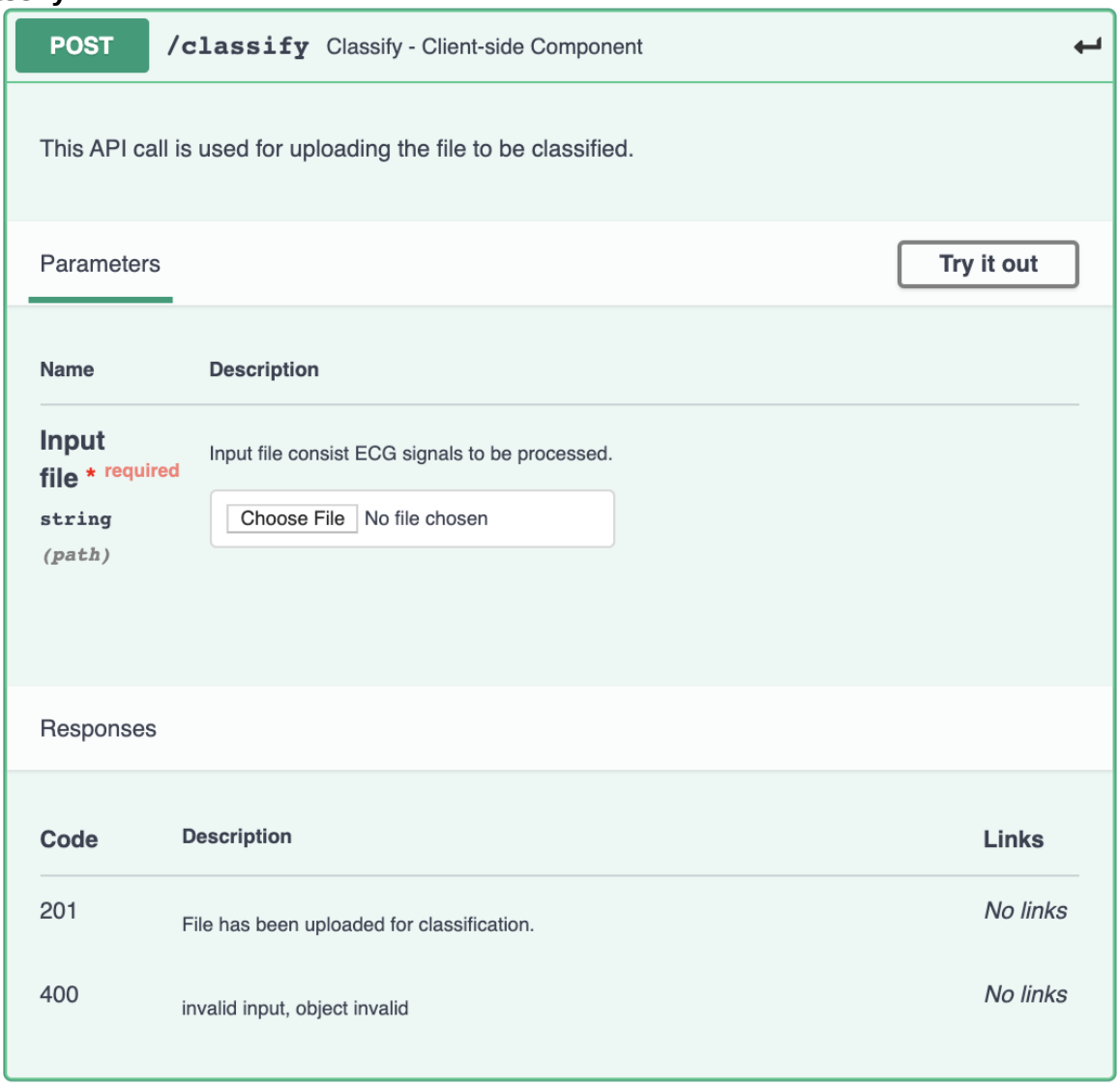
Client-side component API:

Init



Project No. 786767

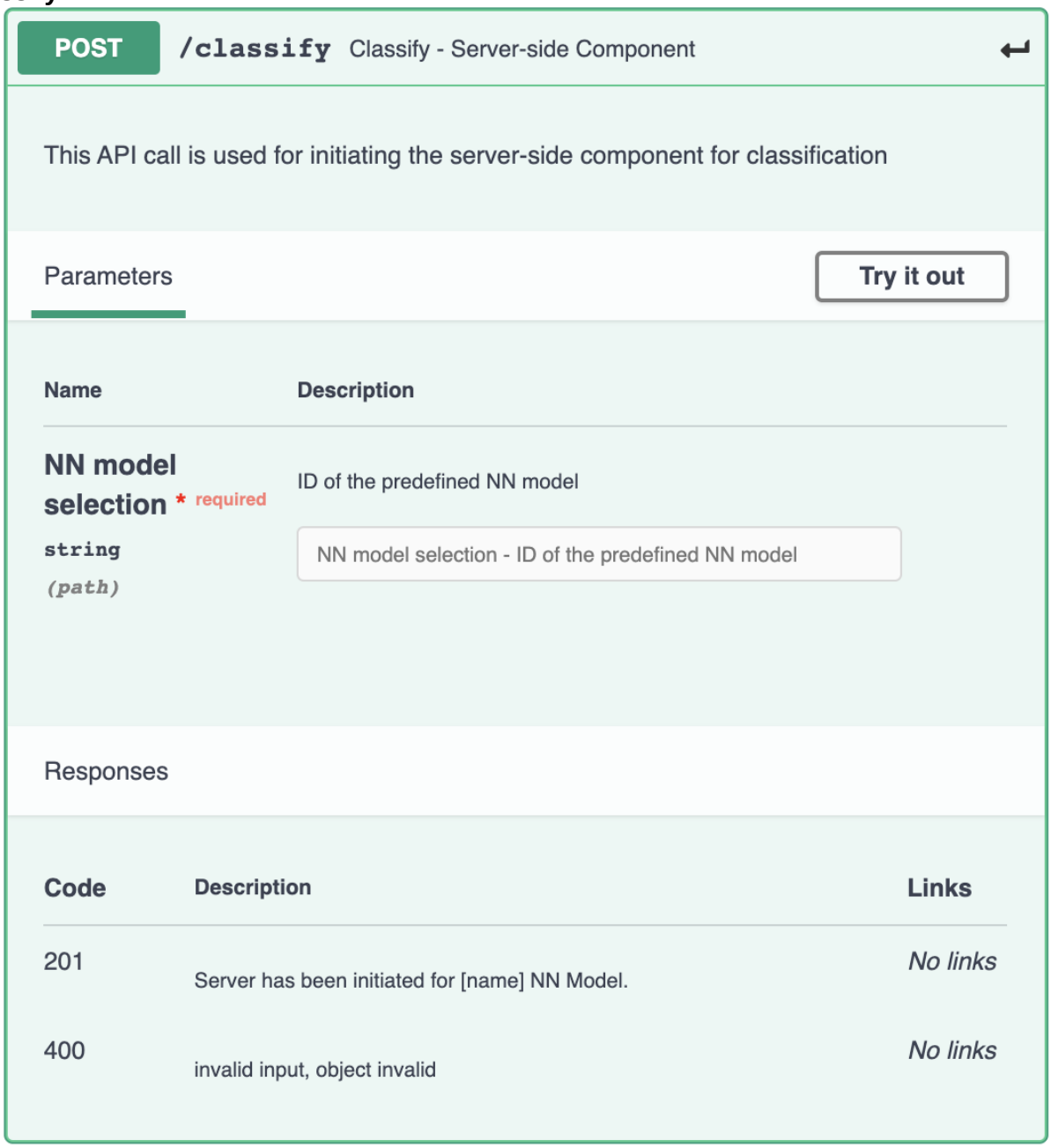
Classify



1.2. Privacy-preserving NN classification based on PHE

Server-side component API:

Classify



Project No. 786767

Client-side component API:

Init

POST `/init` Init - Initialization API Client-side Component

This API call is used for initializing the port numbers and the IP addresses of servers in the client-side component.

Parameters | Try it out

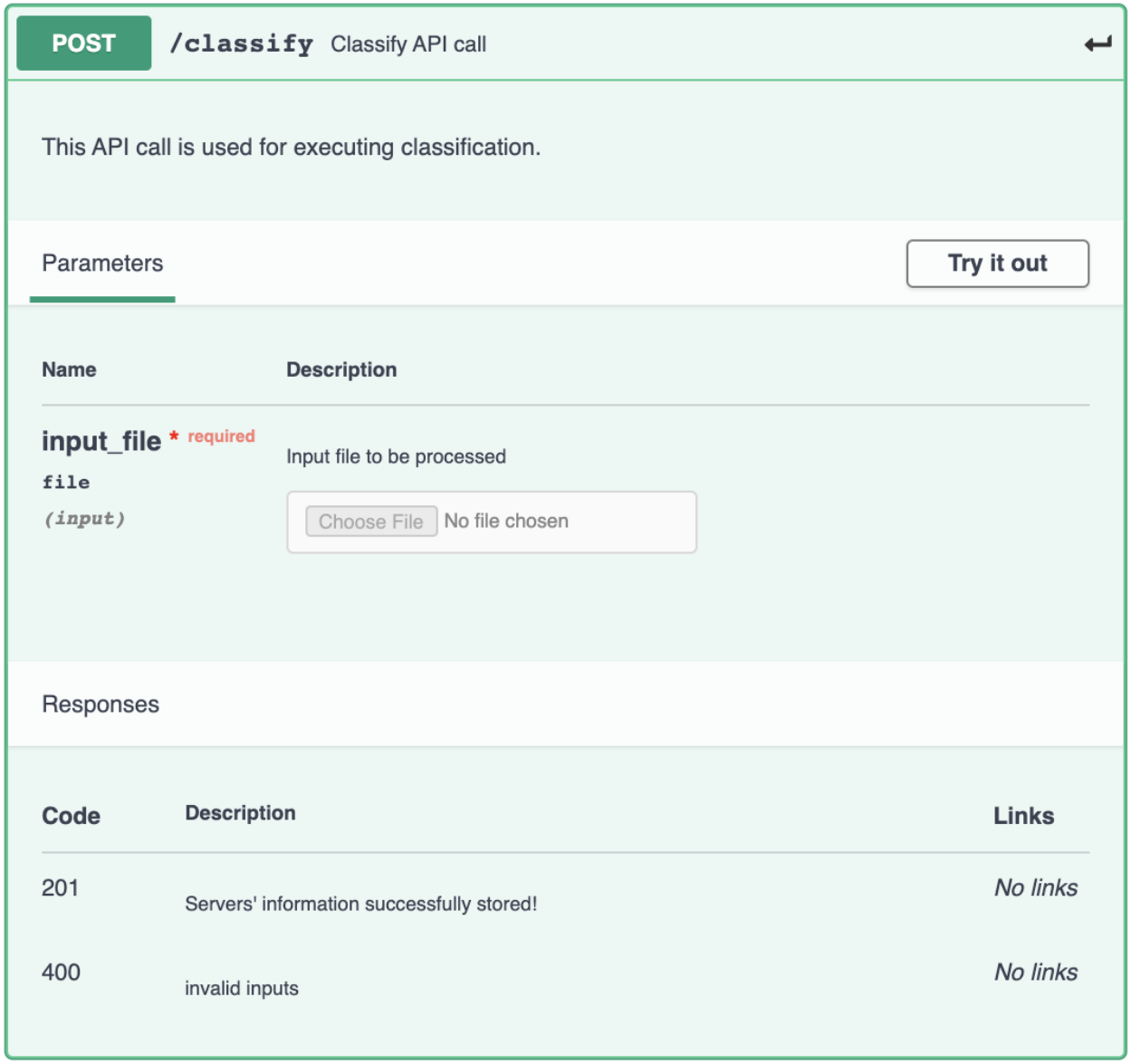
Name	Description
IP_Address * required string *(path)*	IP address of Server IP_Address - IP address of Server
Port_number * required string *(path)*	Port number of Server Port_number - Port number of Server
URL * required string *(path)*	URL of the Server URL - URL of the Server

Responses

Code	Description	Links
201	Servers' information successfully stored!	*No links*
400	invalid inputs	*No links*

Project No. 786767

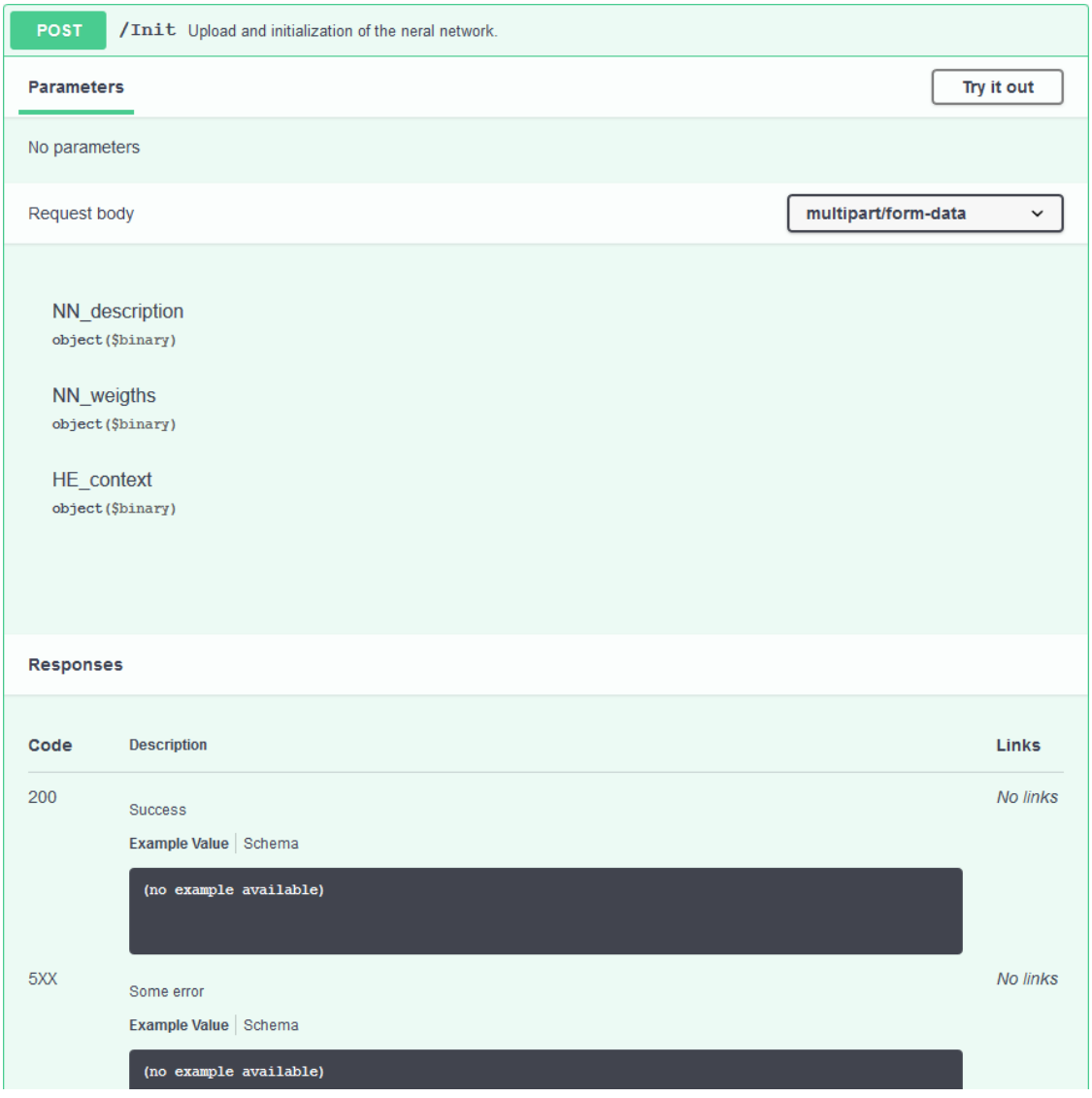
Classify



1.3. Solution based on Homomorphic Encryption

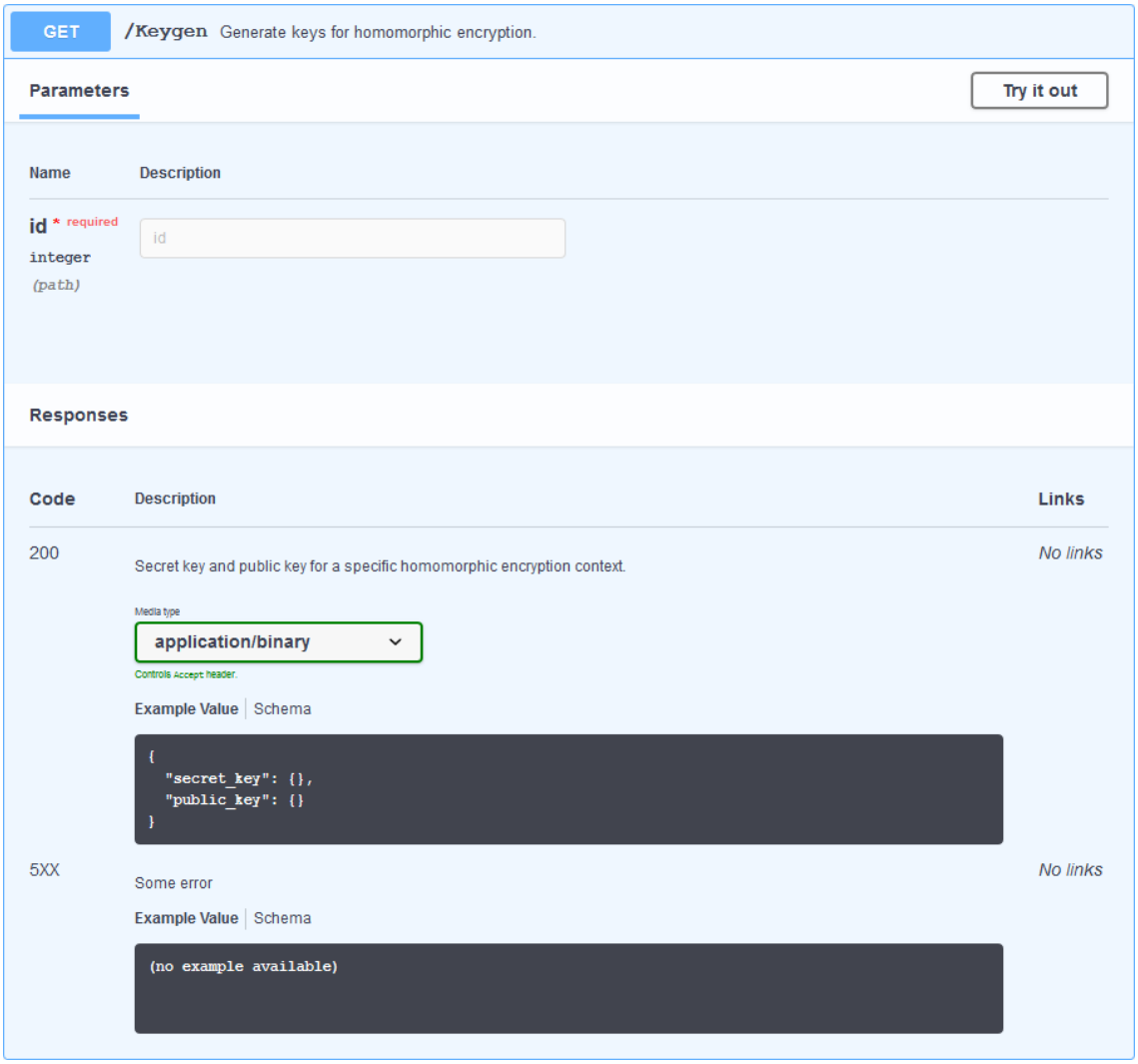
Client-side component API:

Init



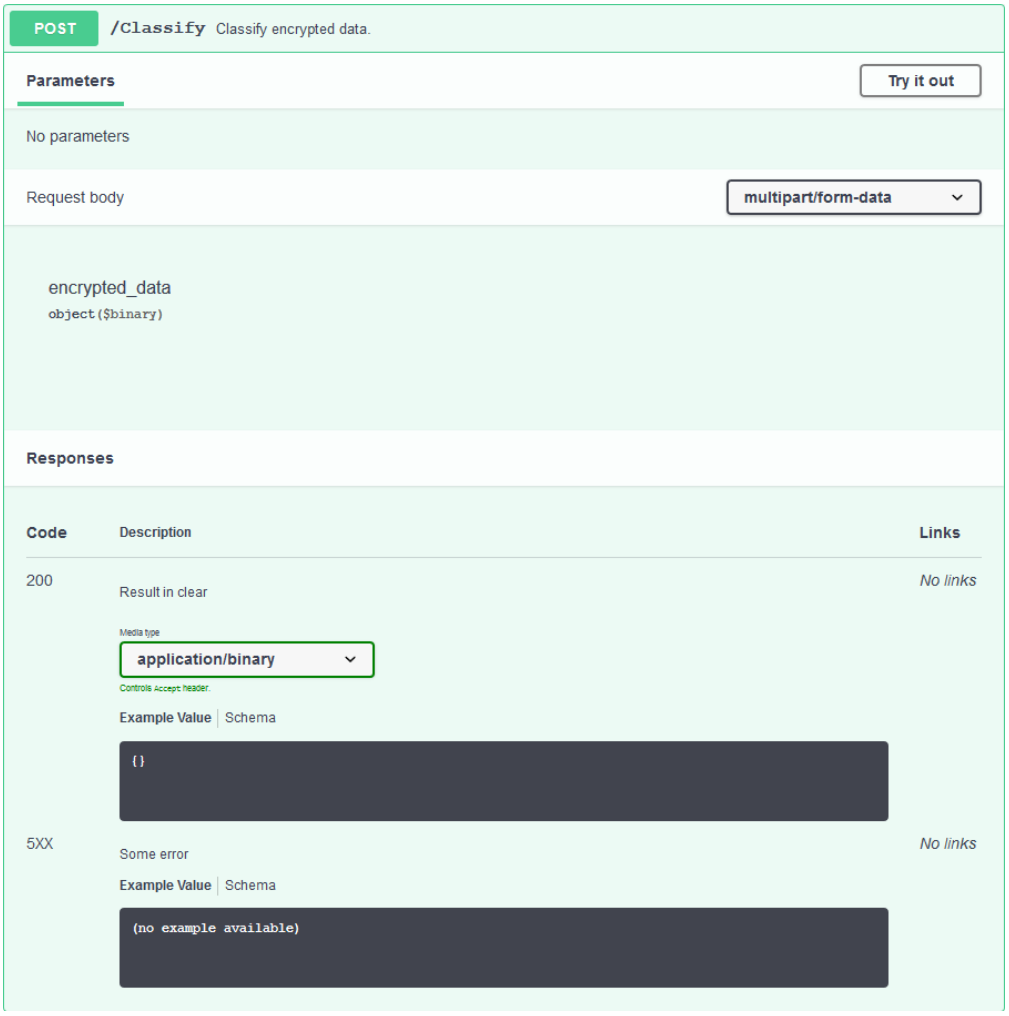
Project No. 786767

Keygen



Project No. 786767

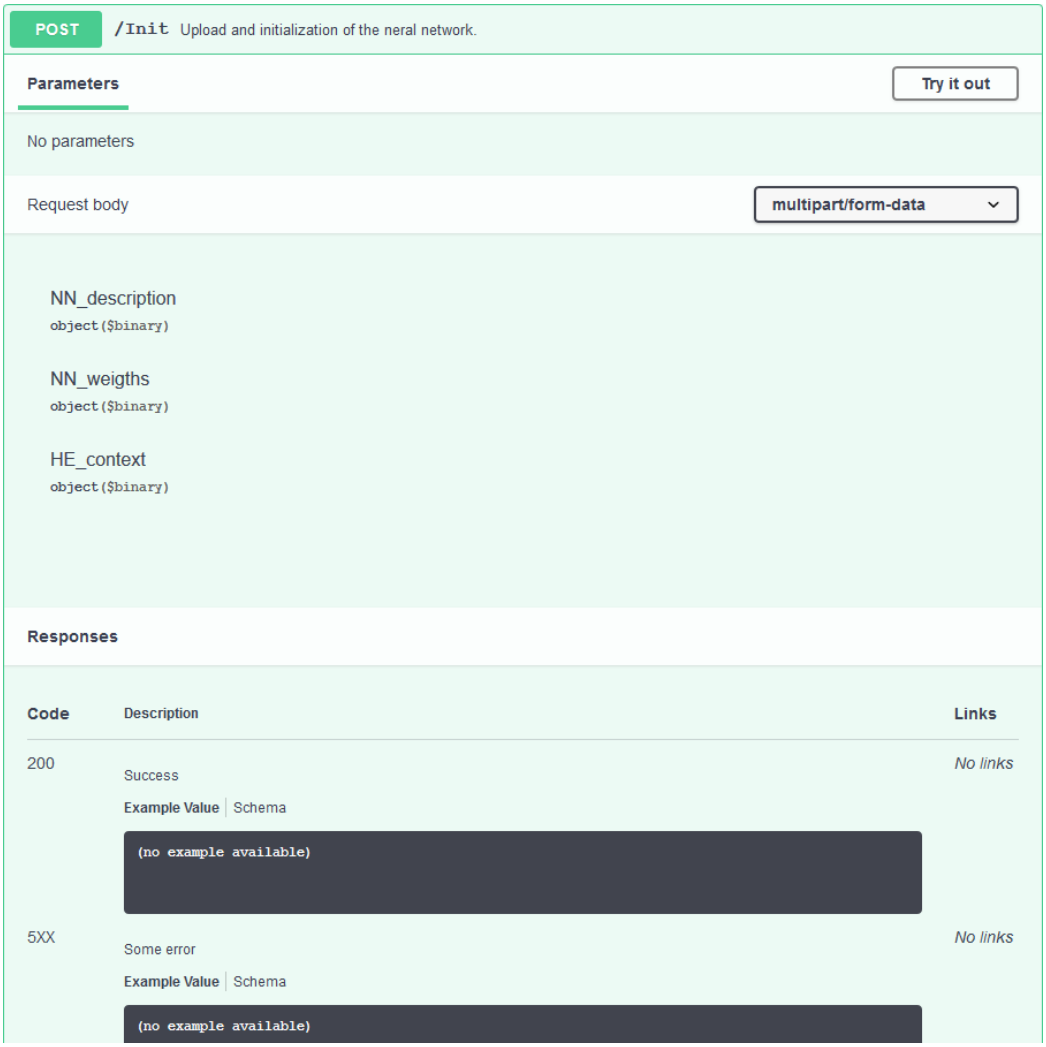
Classify



Project No. 786767

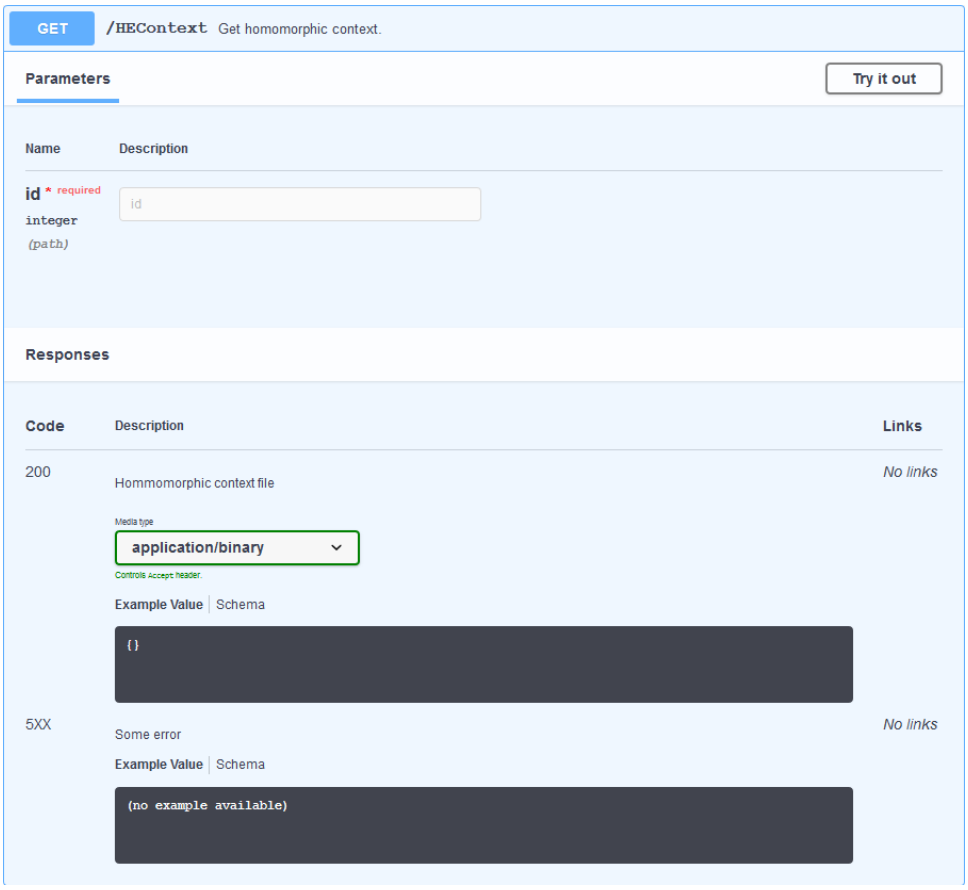
Server-side component API:

Init



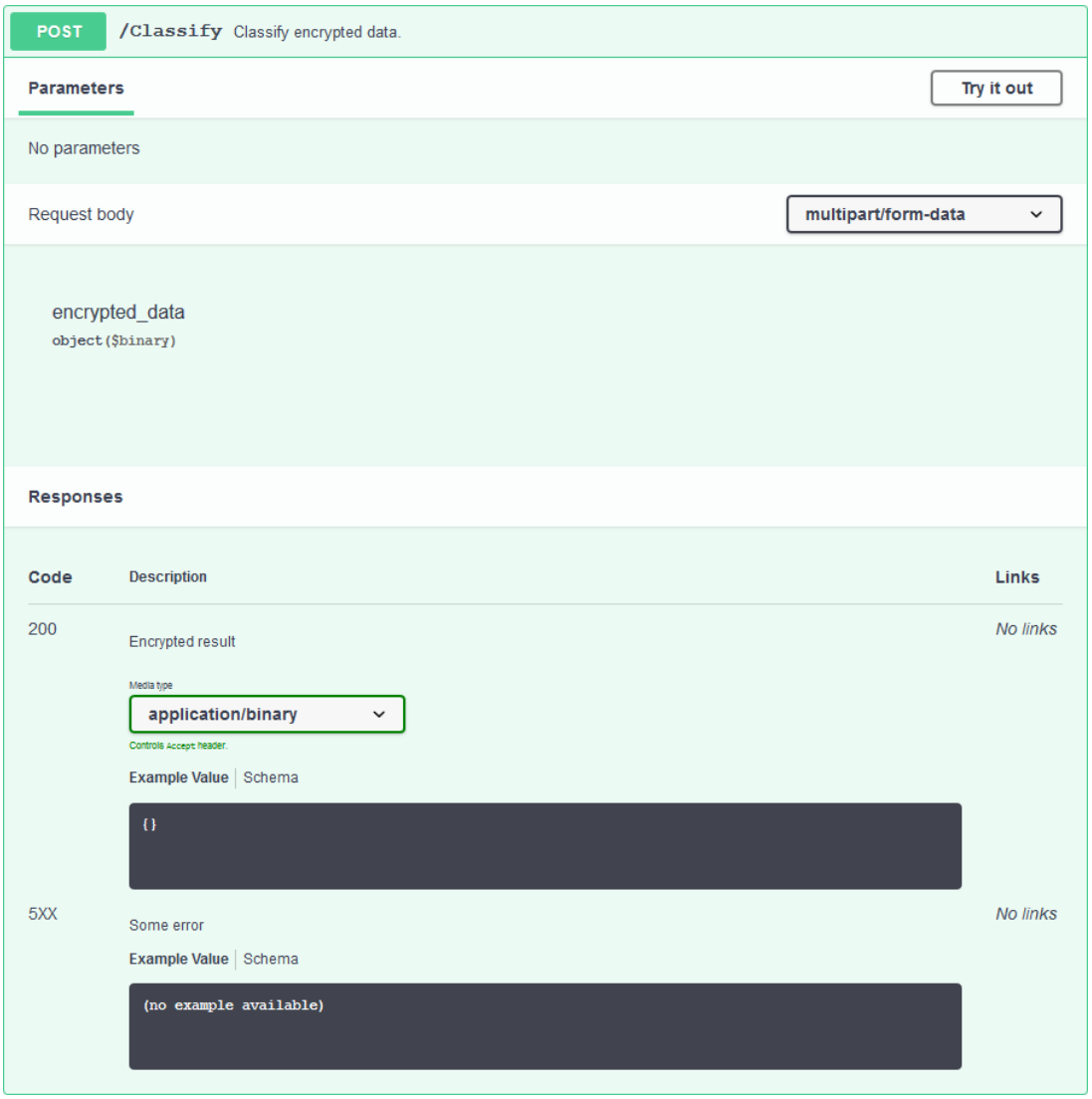
Project No. 786767

HEContext



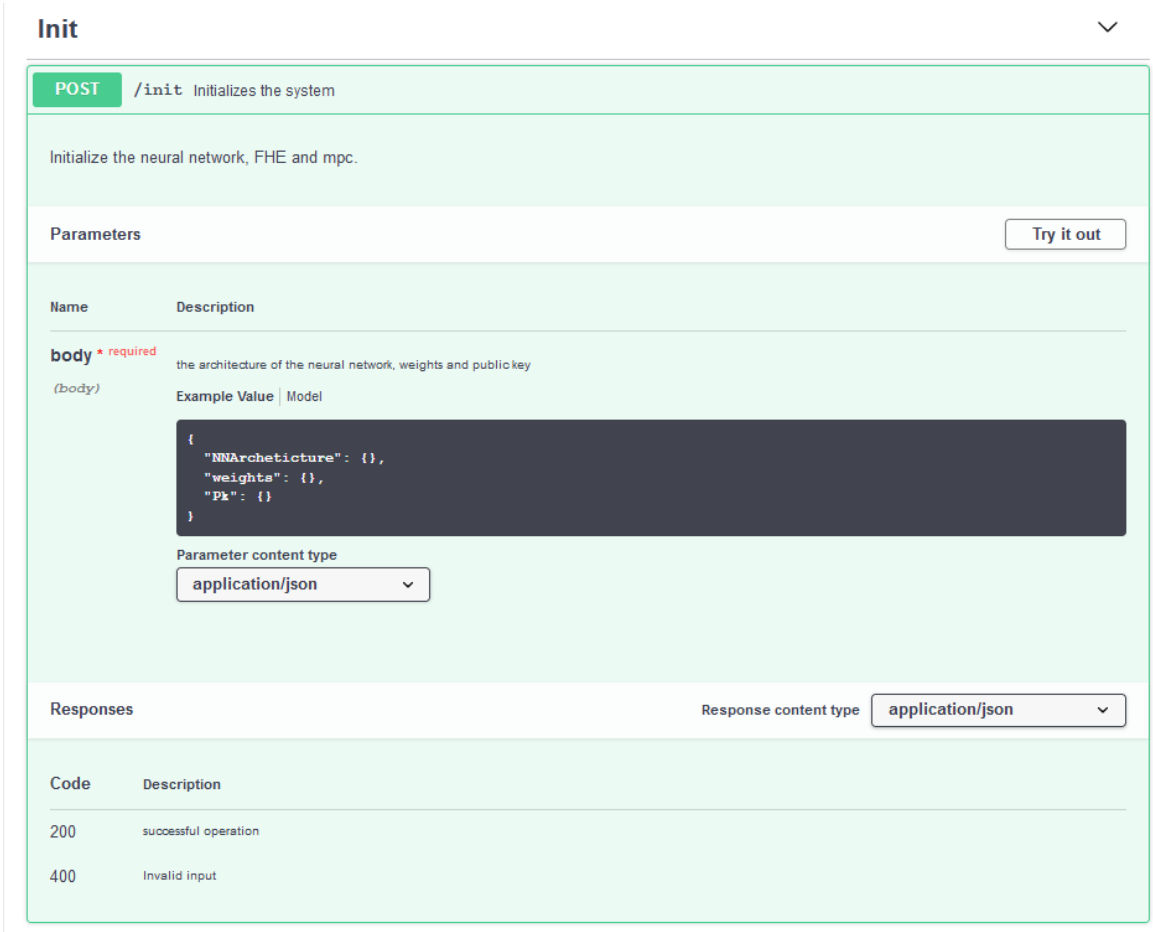
Project No. 786767

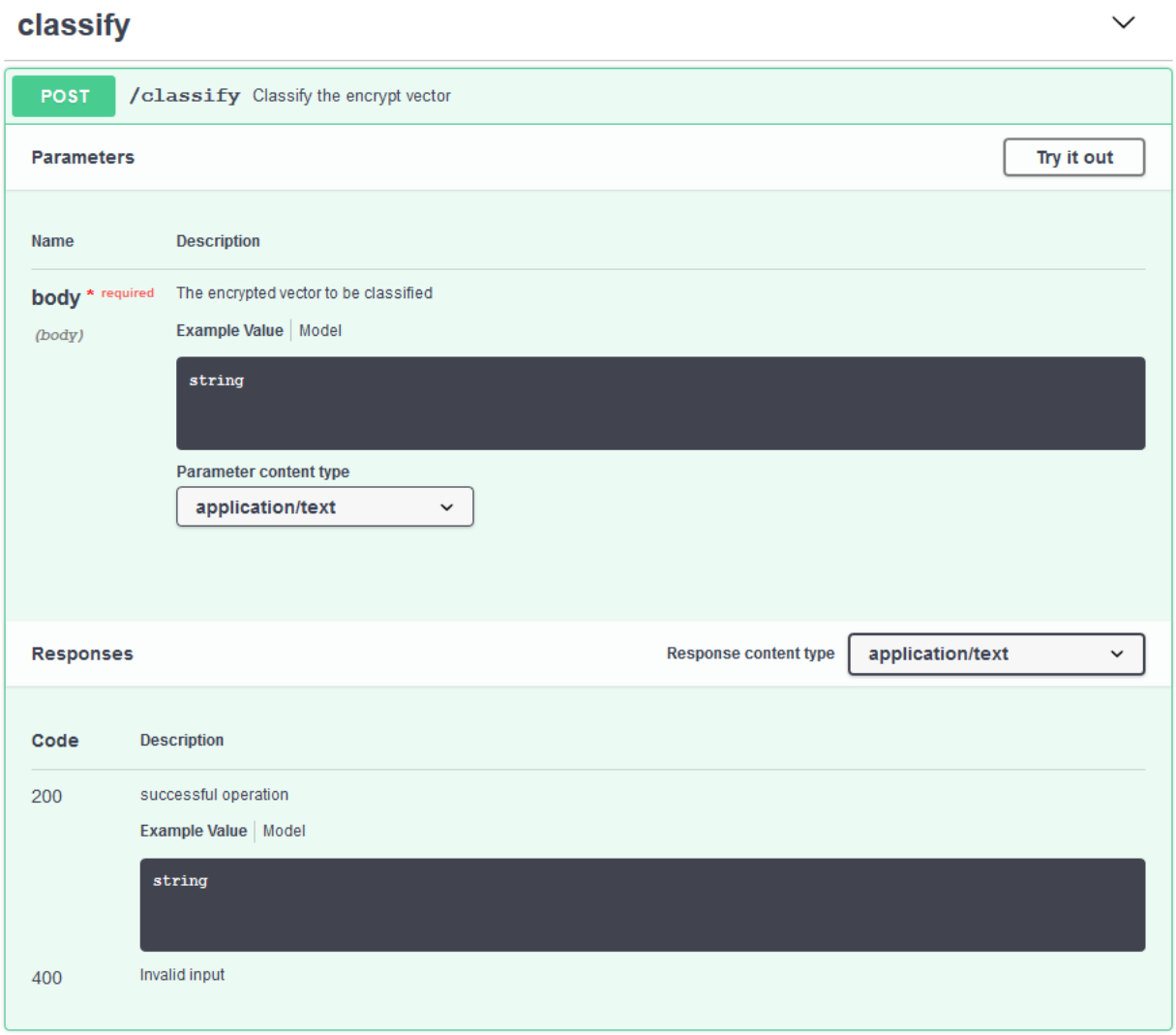
Classify



1.4. Privacy-preserving NN classification based on hybrid approach

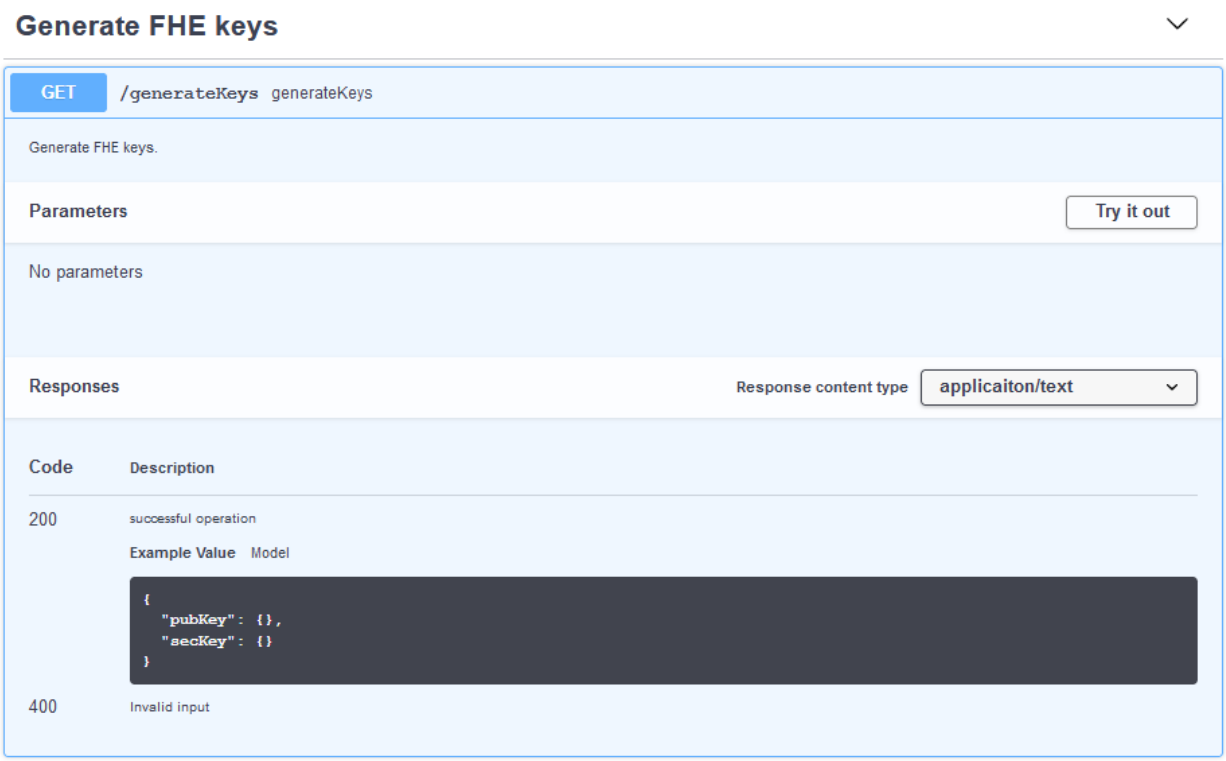
Server-side component API:



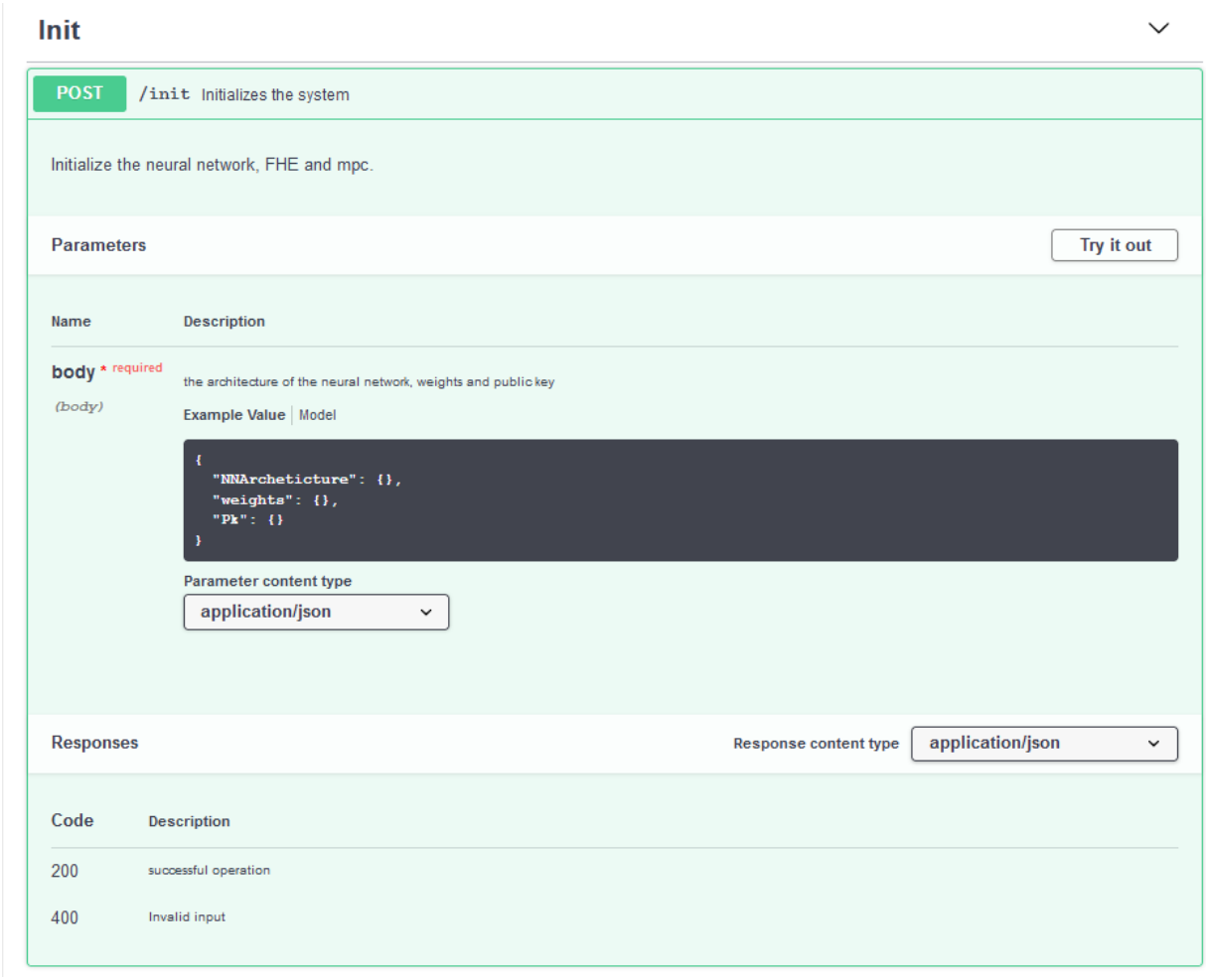

classify
POST
/classify
Classify the encrypt vector
Parameters
Try it out
Name
Description
body * required
The encrypted vector to be classified
(body)
Example Value
Model
string
Parameter content type
application/text
Responses
Response content type
application/text
Code
Description
200
successful operation
Example Value
Model
string
400
Invalid input

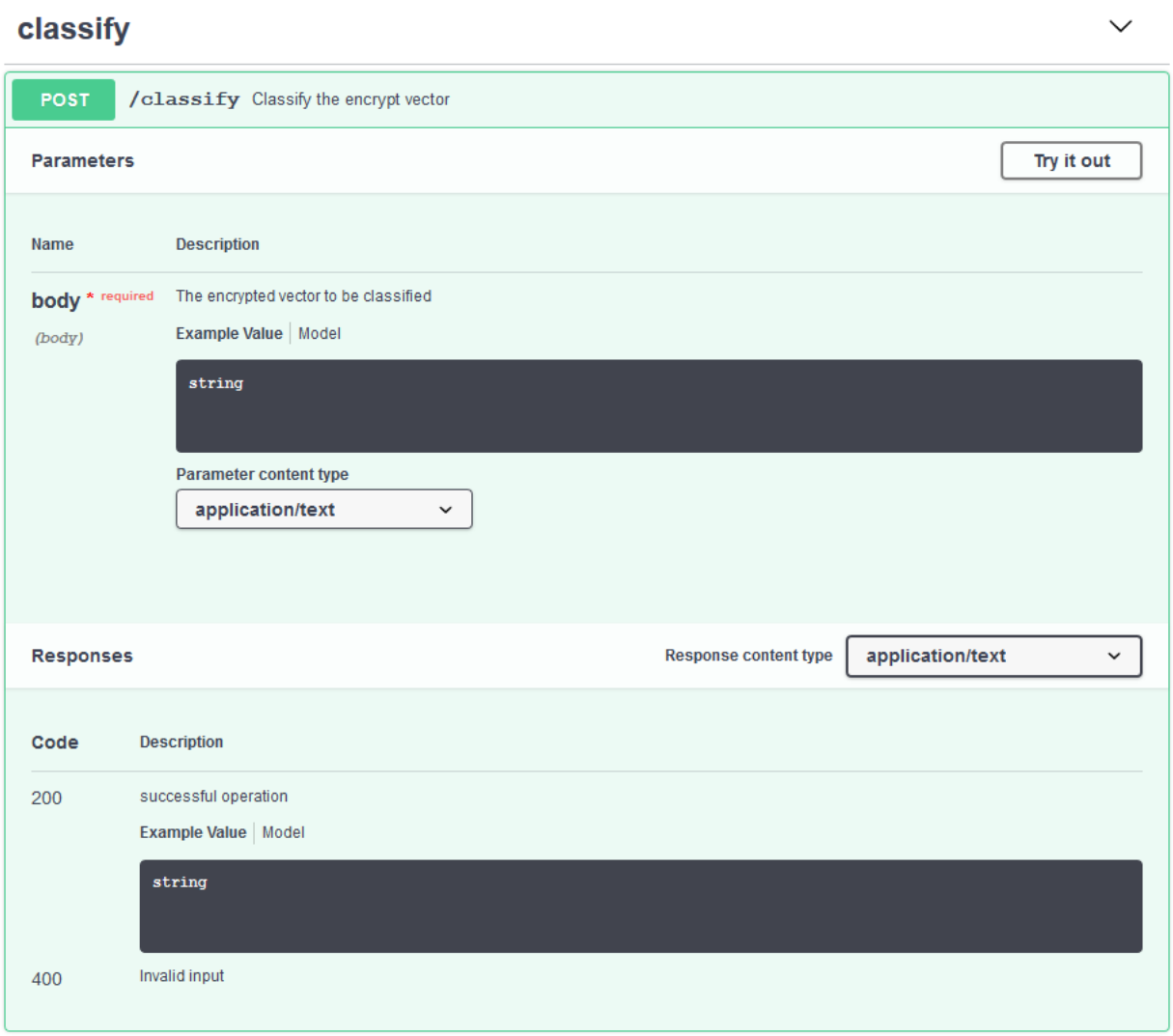
Project No. 786767

Client-side component API:



Project No. 786767



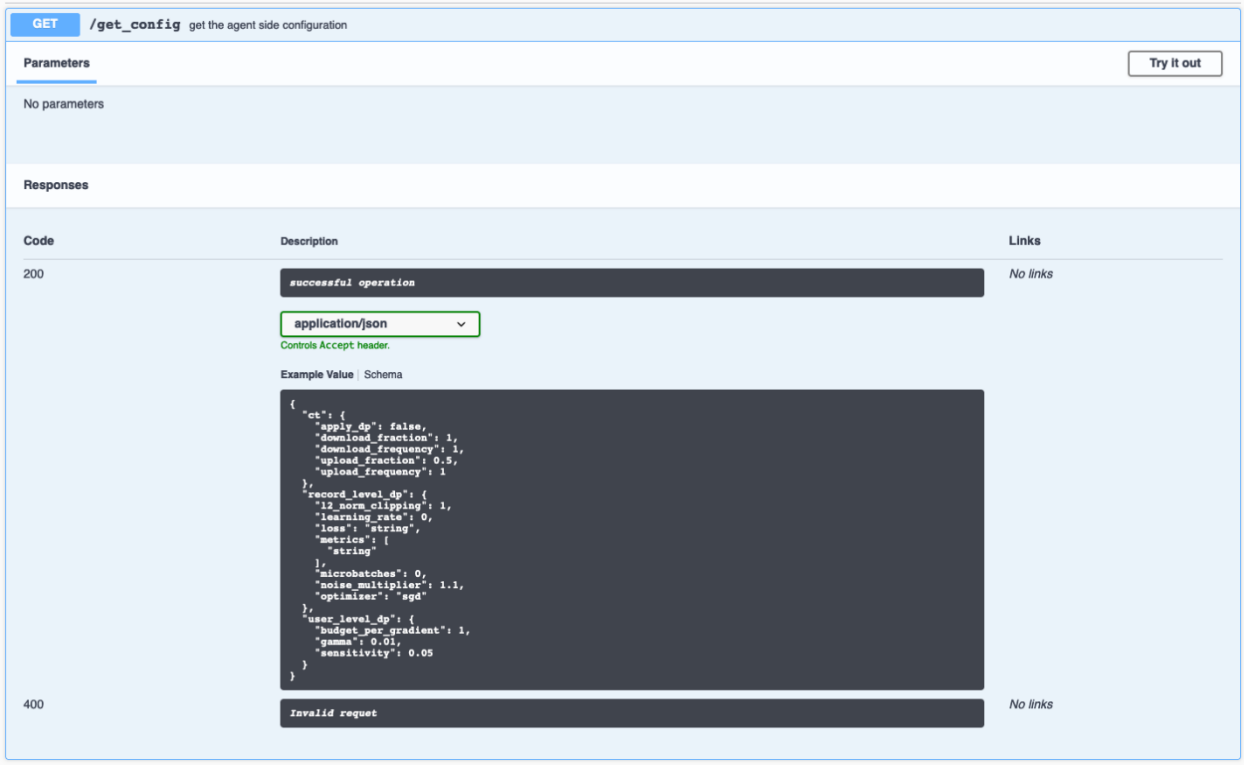

classify
POST
/classify
Classify the encrypt vector
Parameters
Try it out
Name
Description
body * required
(body)
The encrypted vector to be classified
Example Value
Model
string
Parameter content type
application/text
Responses
Response content type
application/text
Code
Description
200
successful operation
Example Value
Model
string
400
Invalid input

Project No. 786767

2. Collaborative Training of Neural Network

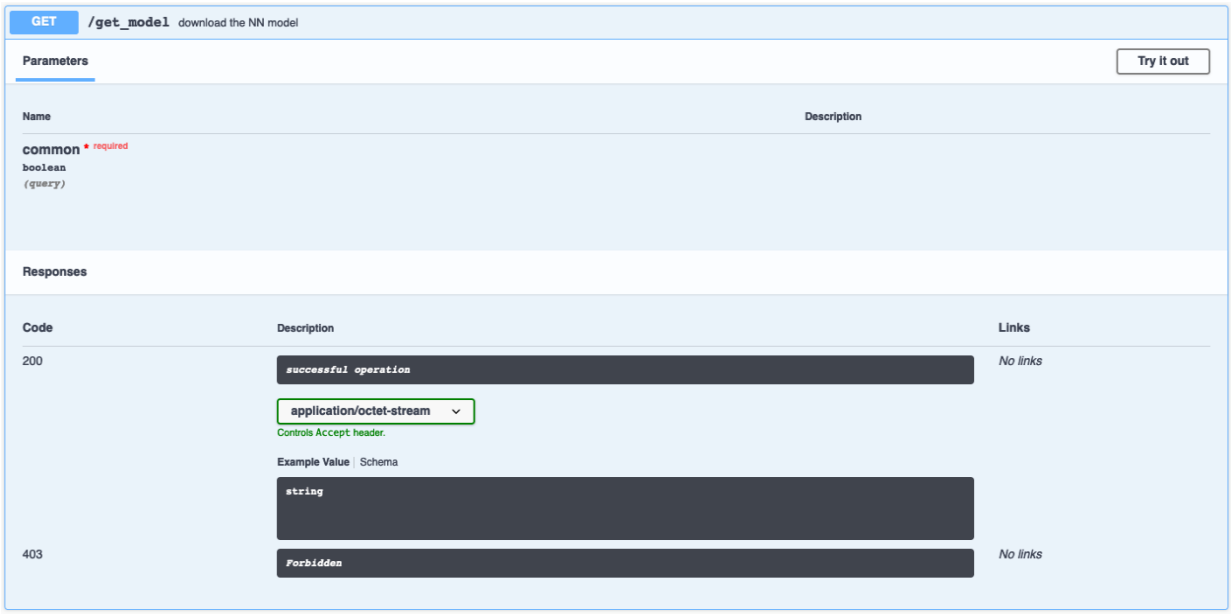
Agent-side API:

Get_config

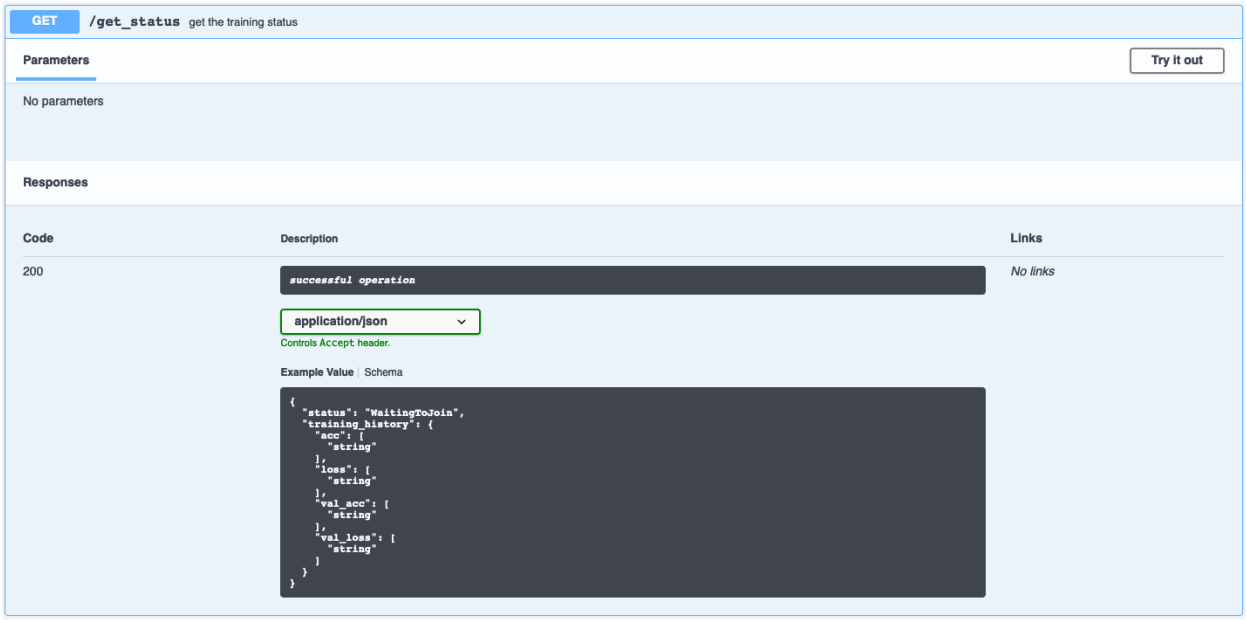


Project No. 786767

Get_model

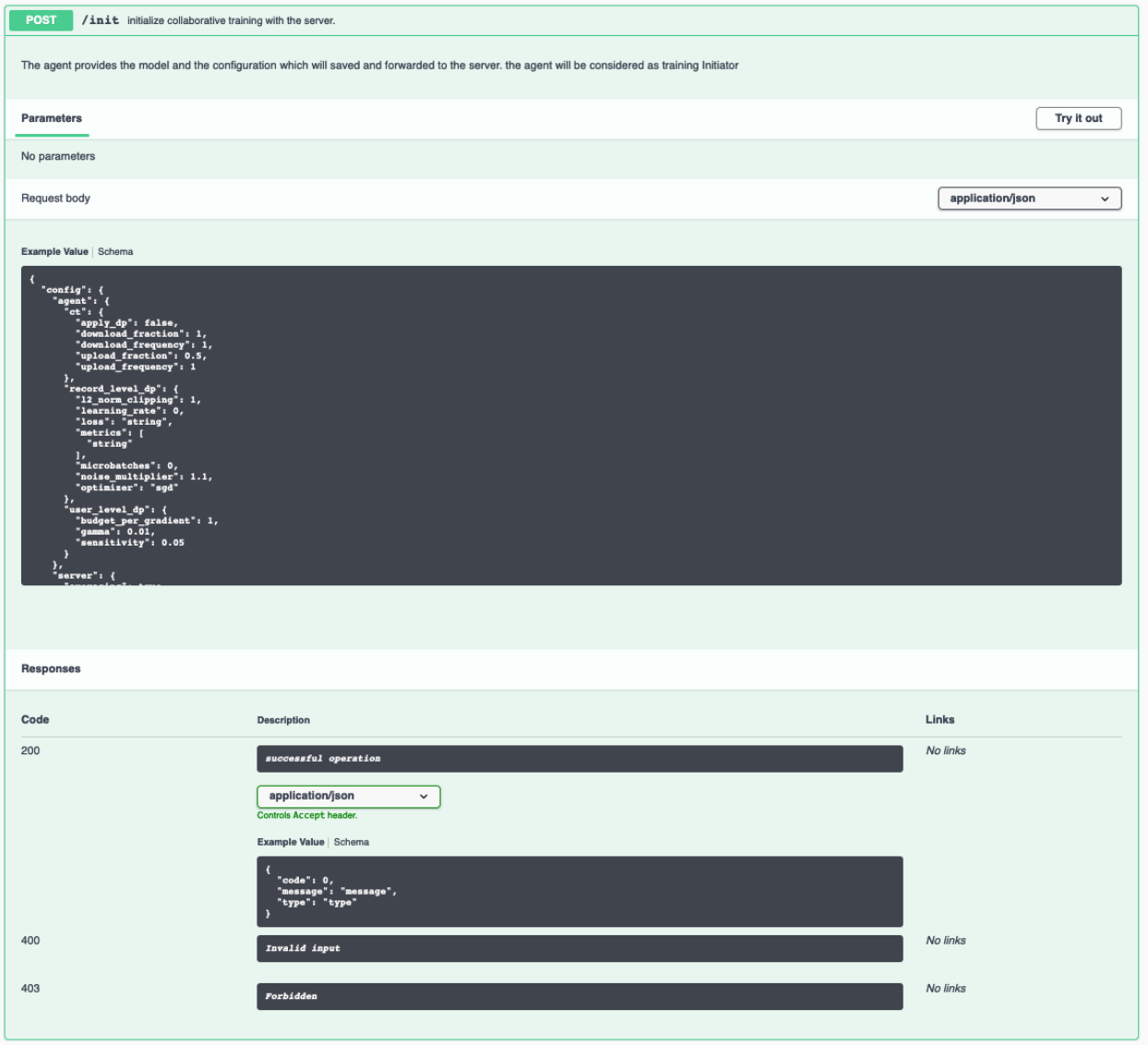


Get_status

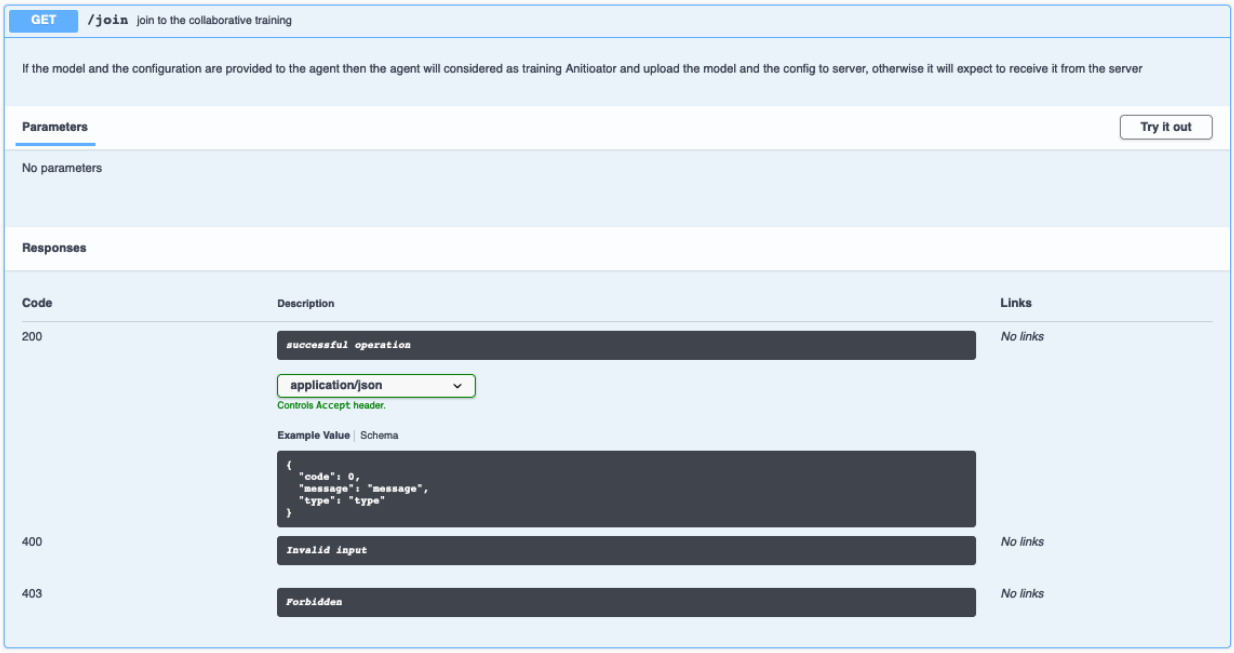


Project No. 786767

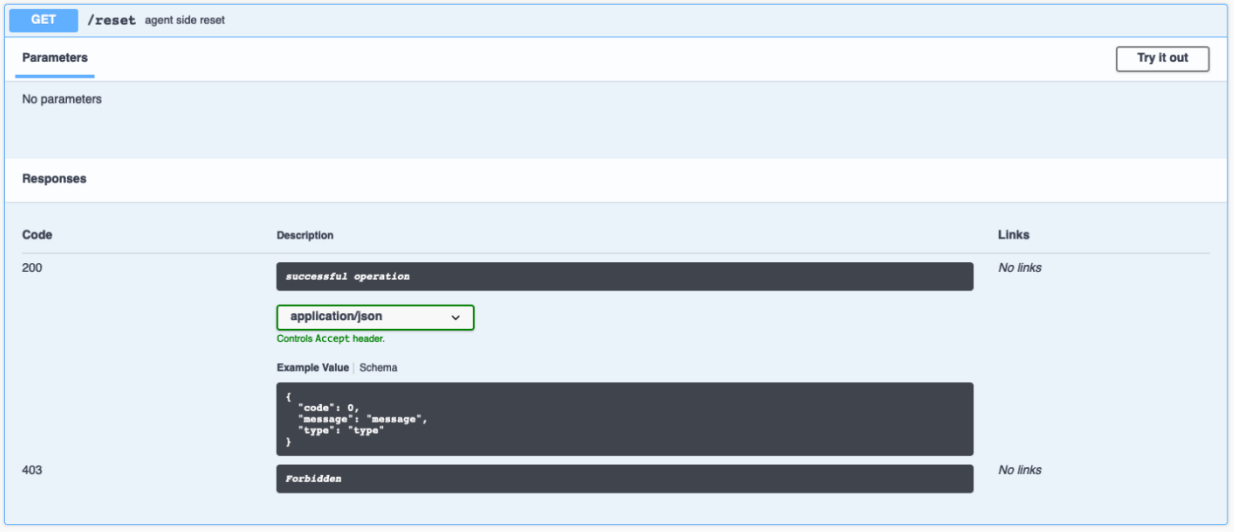
Init



Join

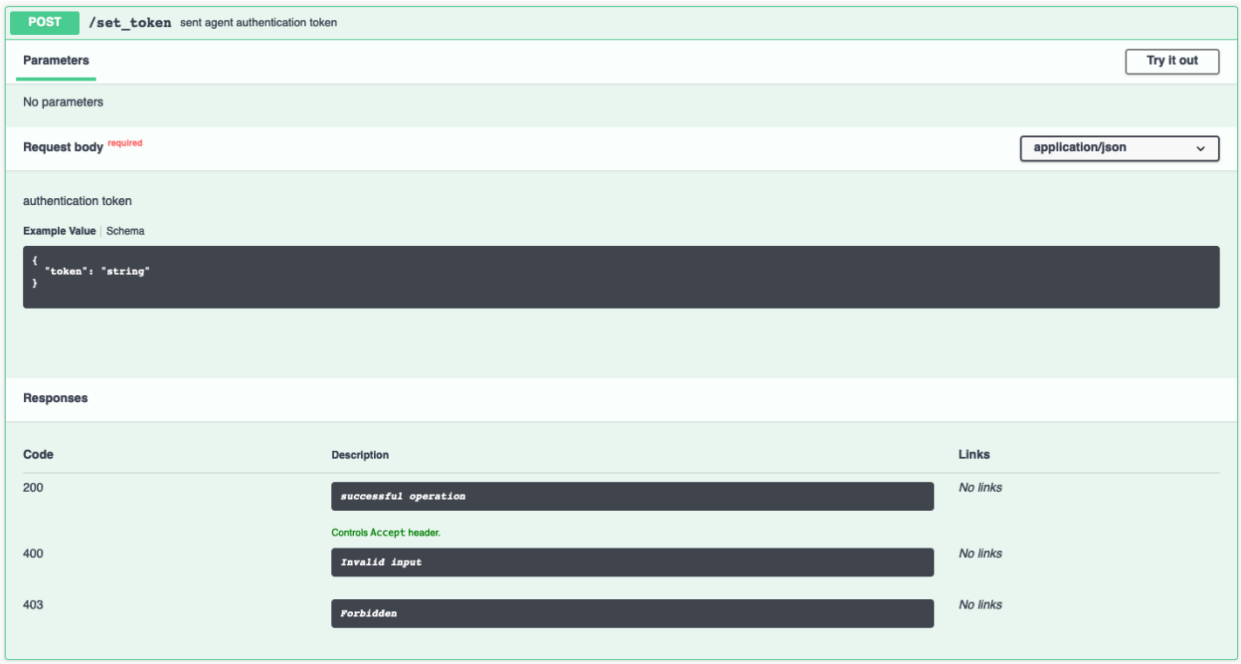

GET /join join to the collaborative training
If the model and the configuration are provided to the agent then the agent will considered as training Anitioator and upload the model and the config to server, otherwise it will expect to receive it from the server
Parameters
Try it out
No parameters
Responses
Code
Description
Links
200
successful operation
No links
application/json
Controls Accept header.
Example Value | Schema
{
"code": 0,
"message": "message",
"type": "type"
}
400
Invalid input
No links
403
Forbidden
No links

Reset


GET /reset agent side reset
Parameters
Try it out
No parameters
Responses
Code
Description
Links
200
successful operation
No links
application/json
Controls Accept header.
Example Value | Schema
{
"code": 0,
"message": "message",
"type": "type"
}
403
Forbidden
No links

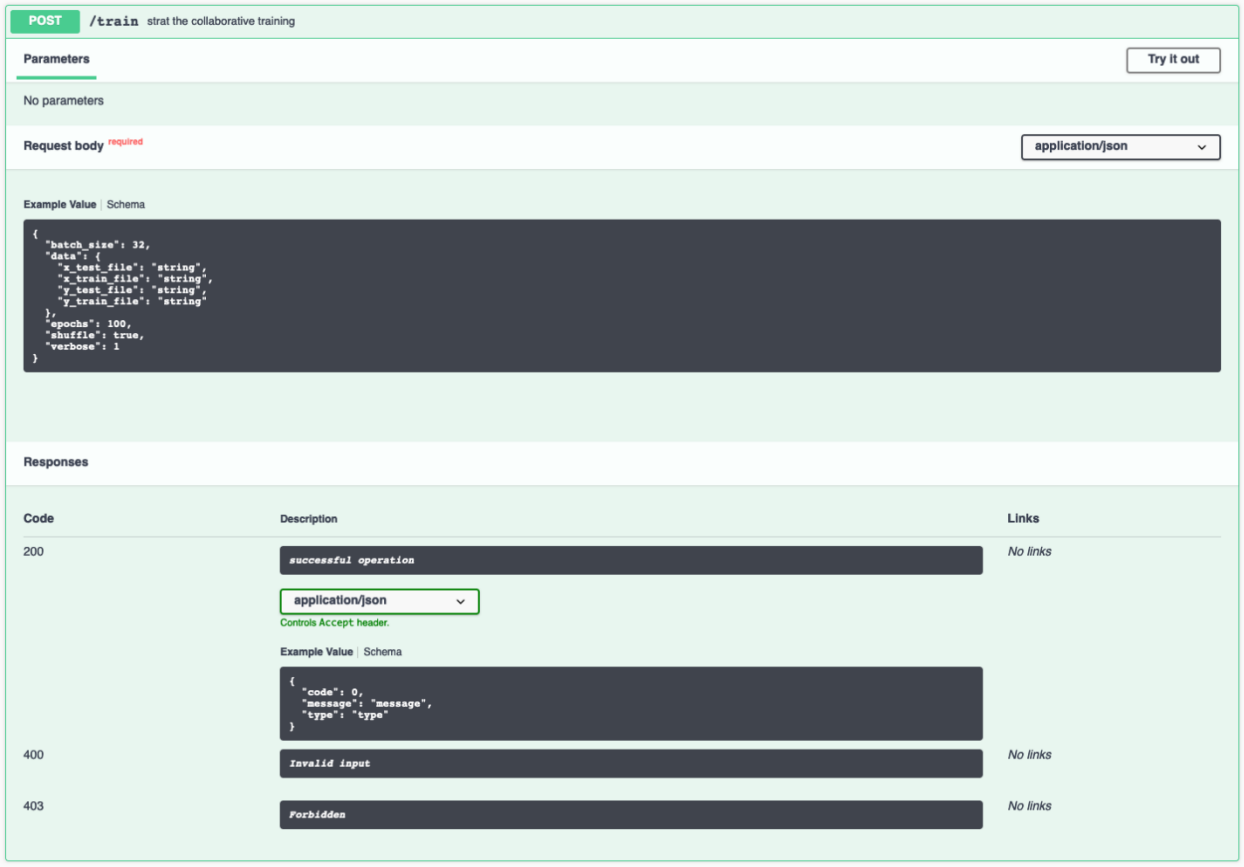
Project No. 786767

Set_token



Project No. 786767

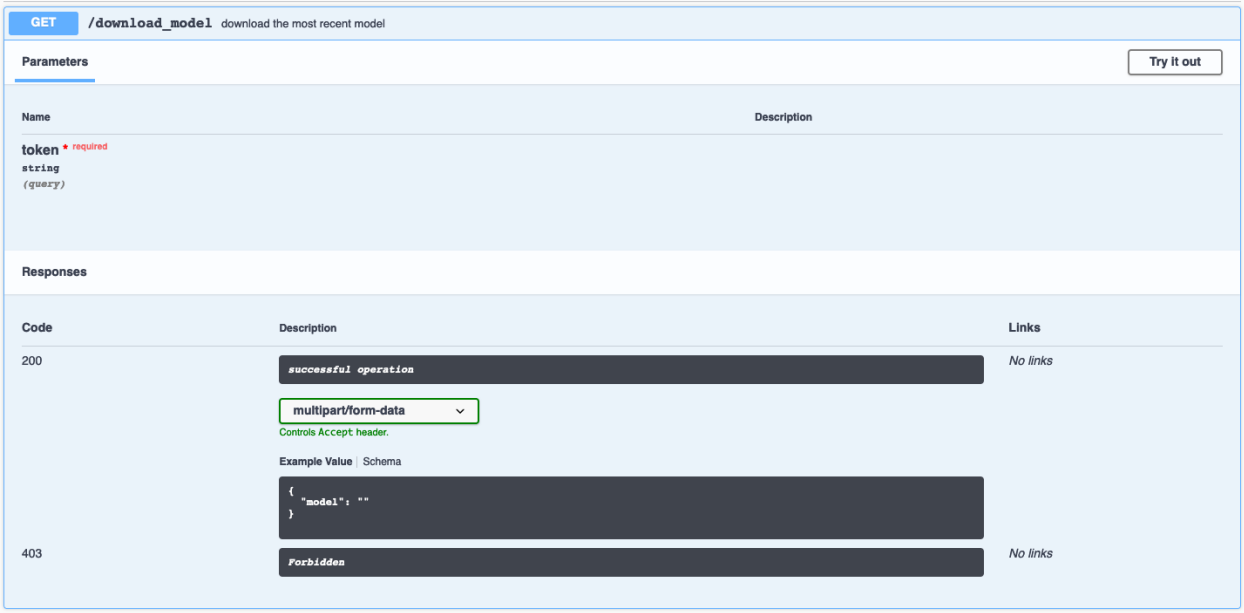
Train

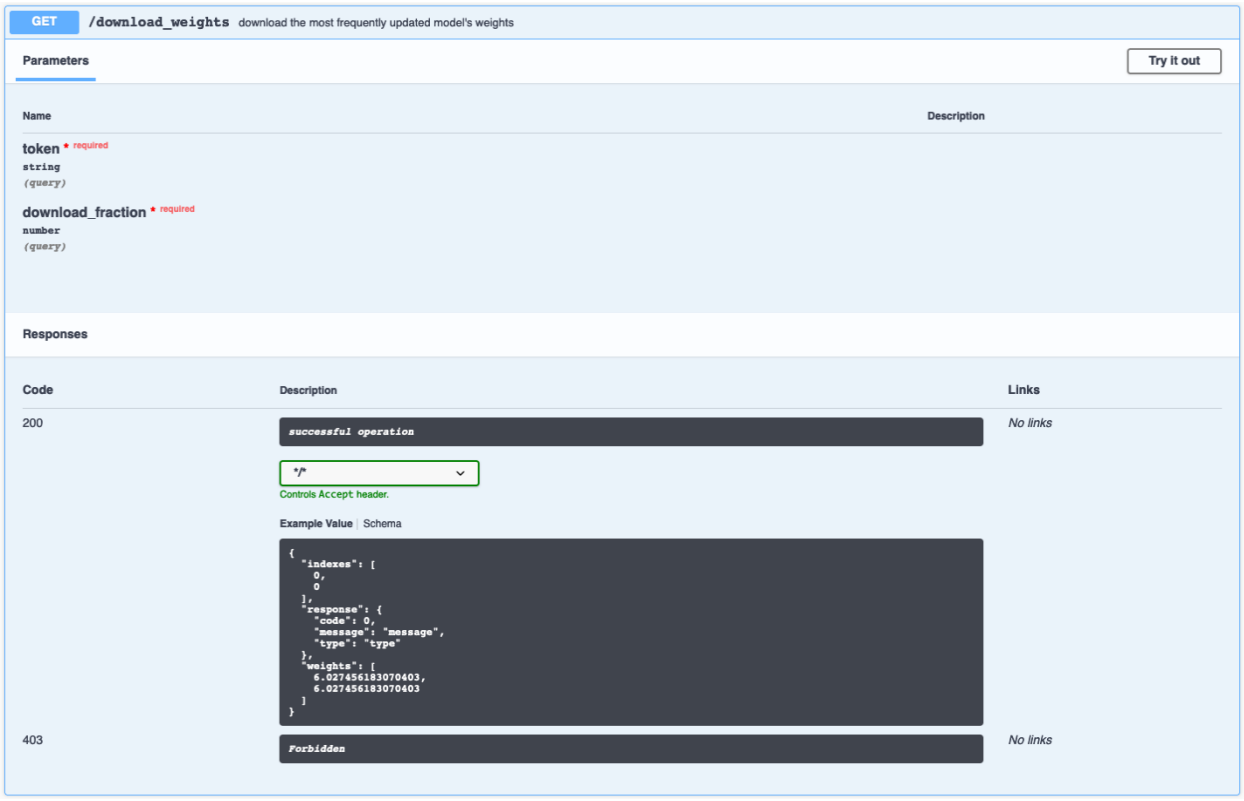


Project No. 786767

Server-side API:

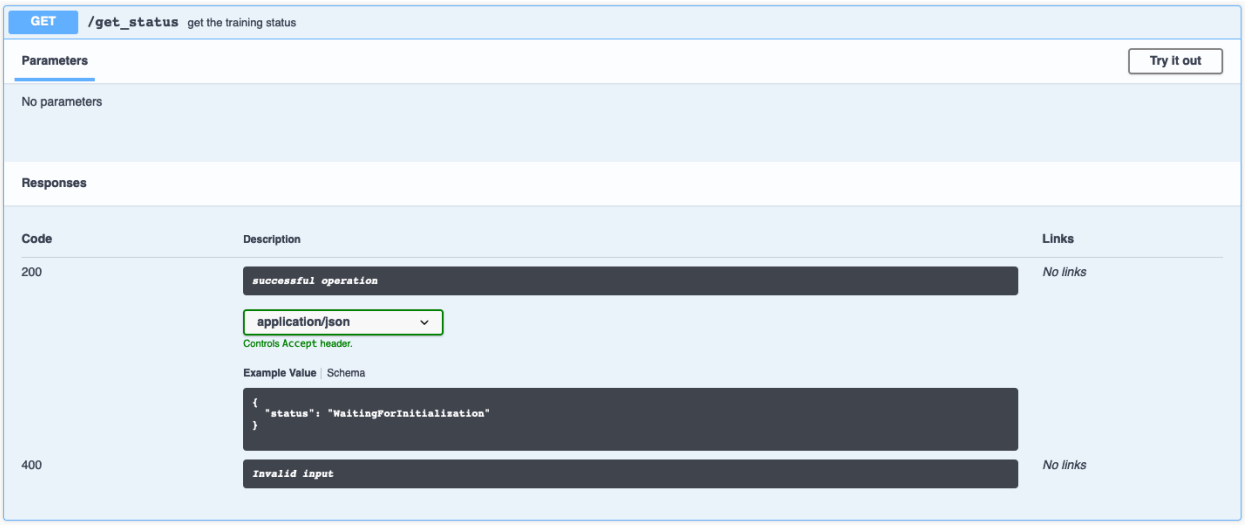
Download_model





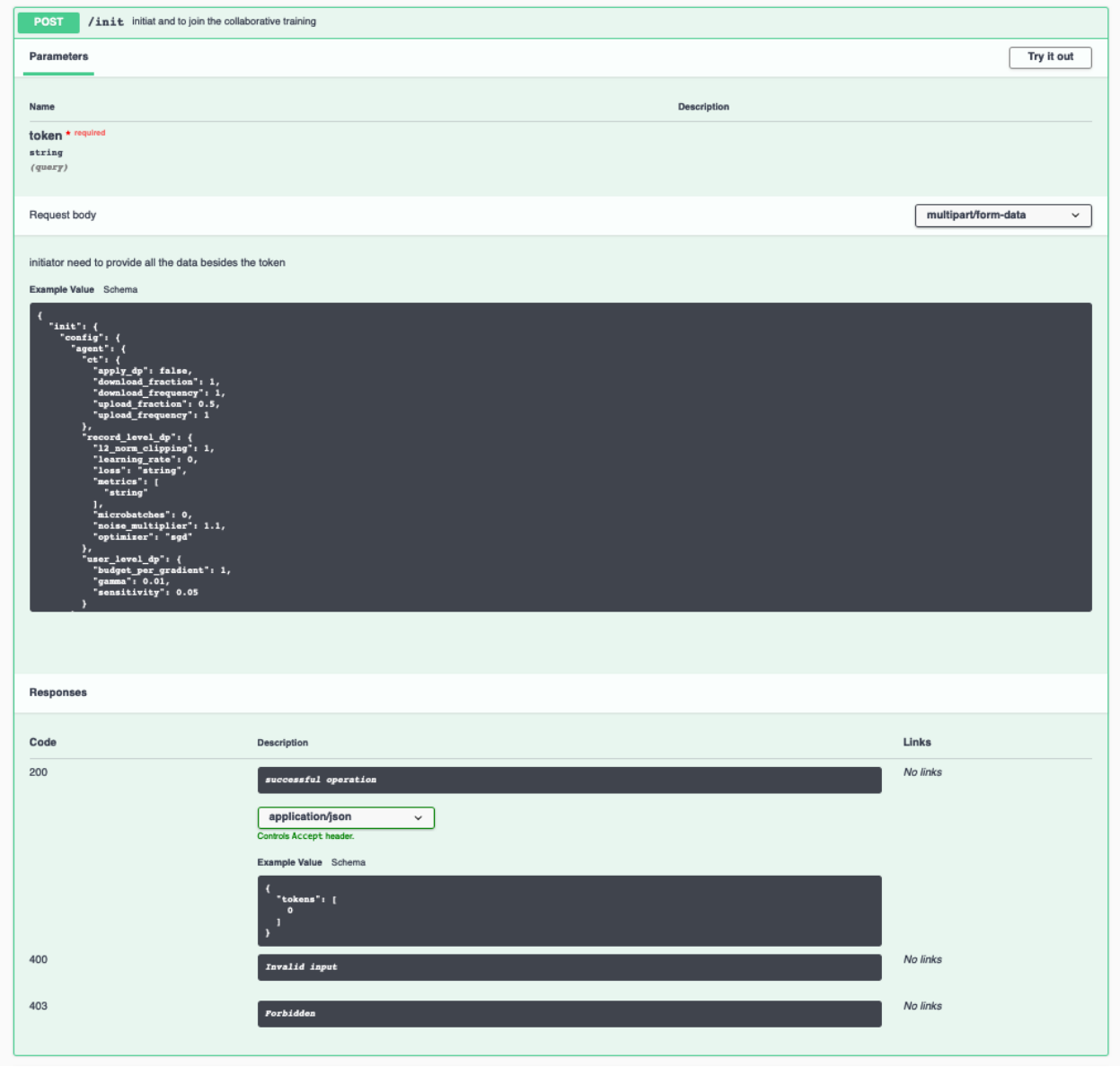
Project No. 786767

Get_status



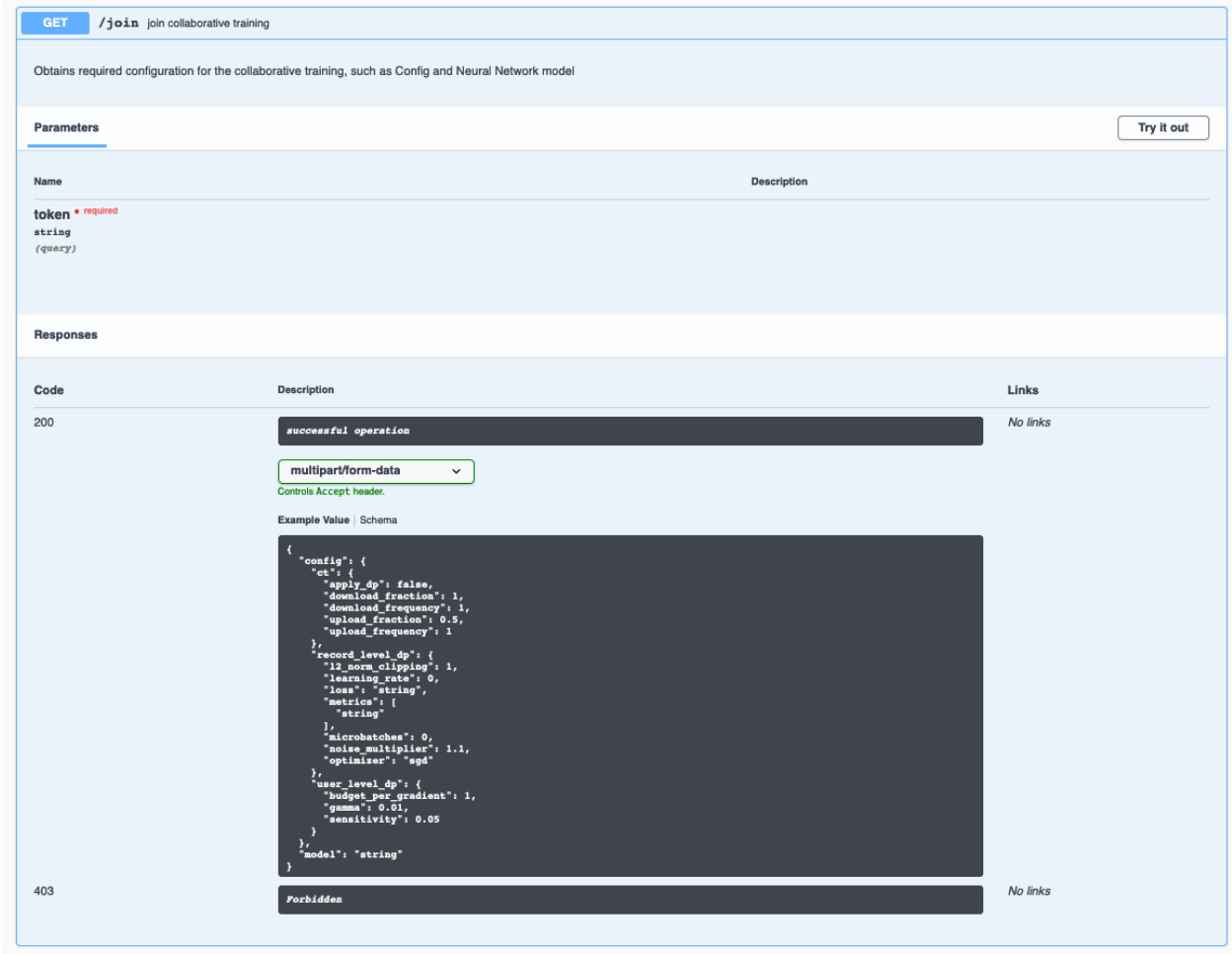
Project No. 786767

Init



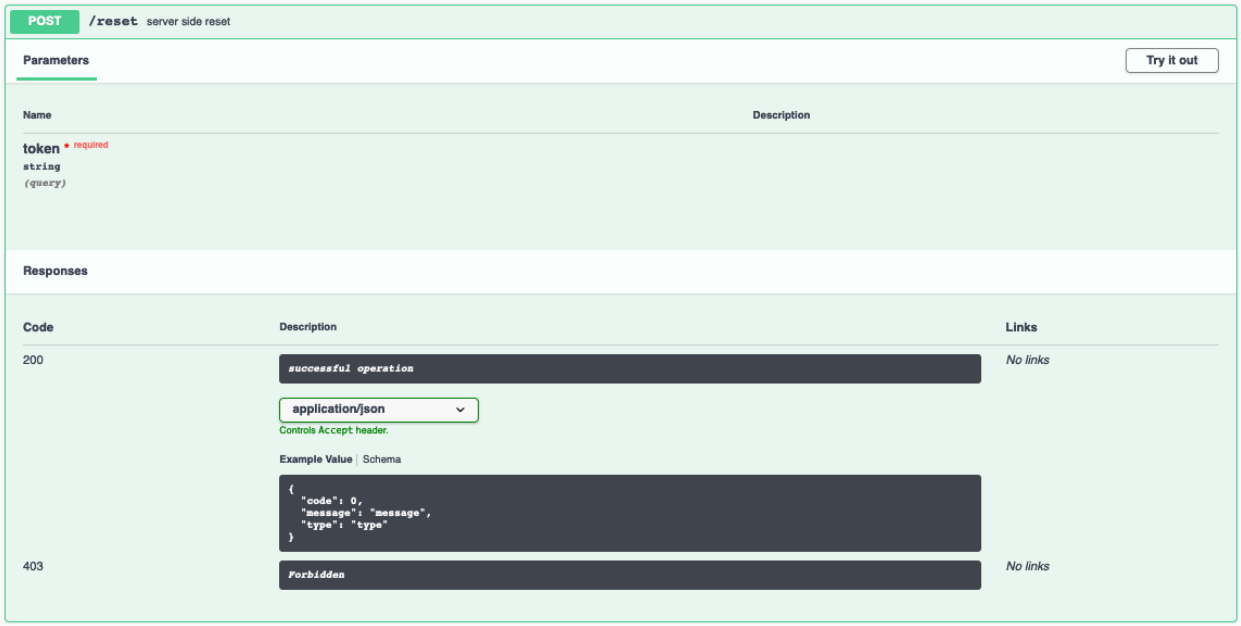
Project No. 786767

Join

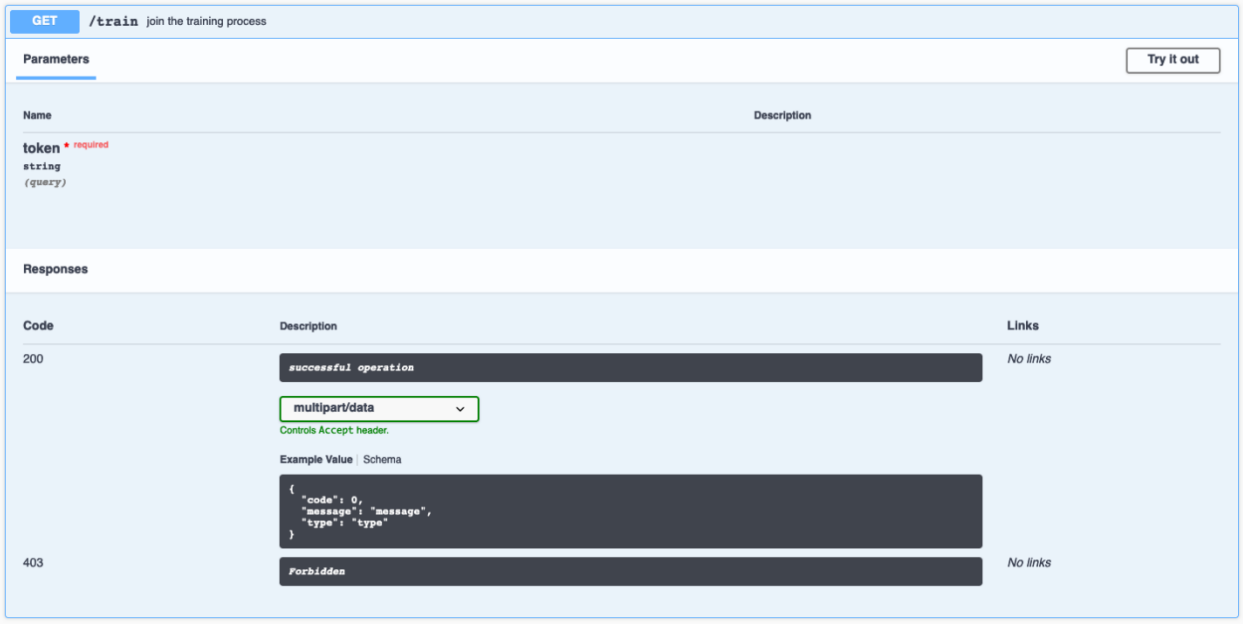


Project No. 786767

Reset

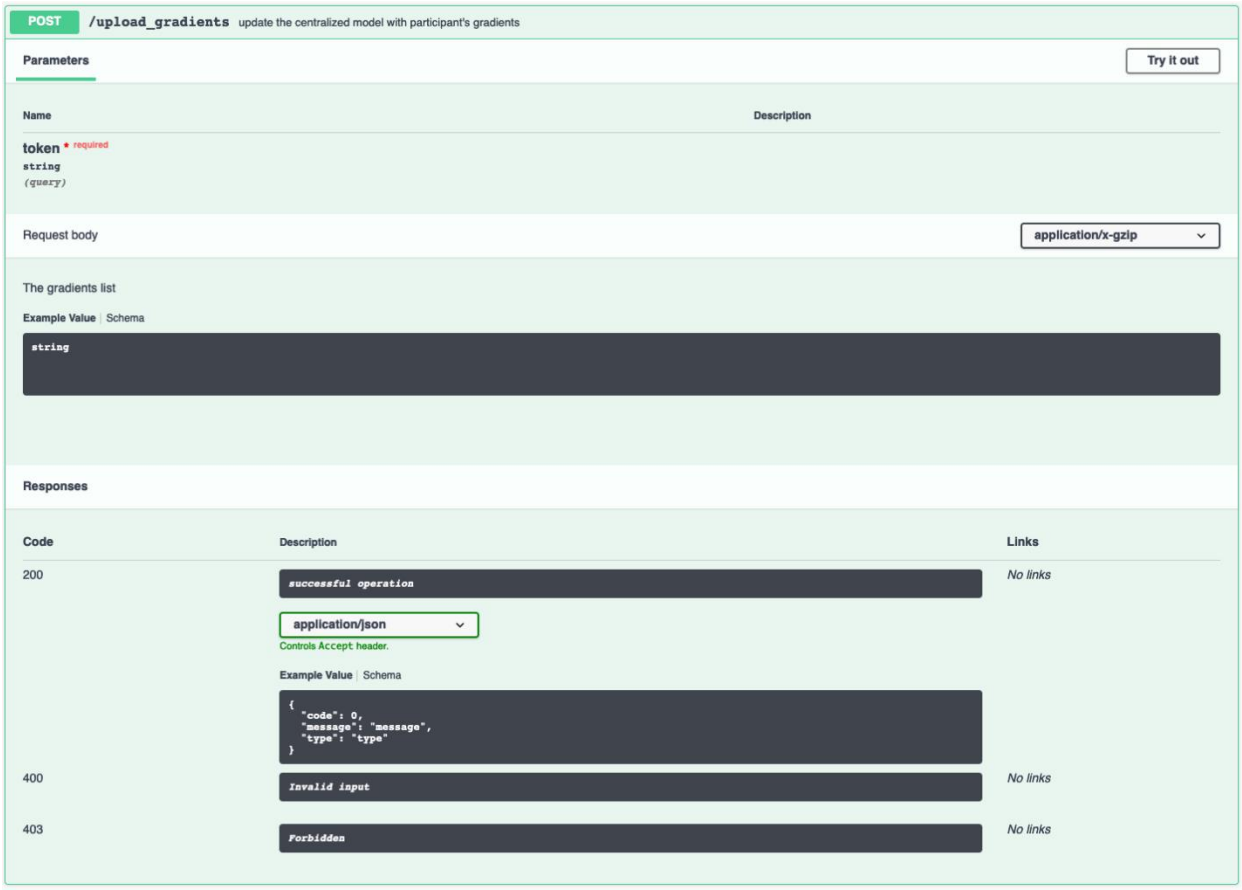


Train



Project No. 786767

Upload_gradients



3. Clustering

3.1. Privacy-preserving clustering based on 2PC

Server-side component API:

Start

POST `/start` Start - Start Clustering for both parties

This API call is used for starting the clustering algorithm in client-server mode and two non-colluding servers mode.

Parameters | Try it out

No parameters

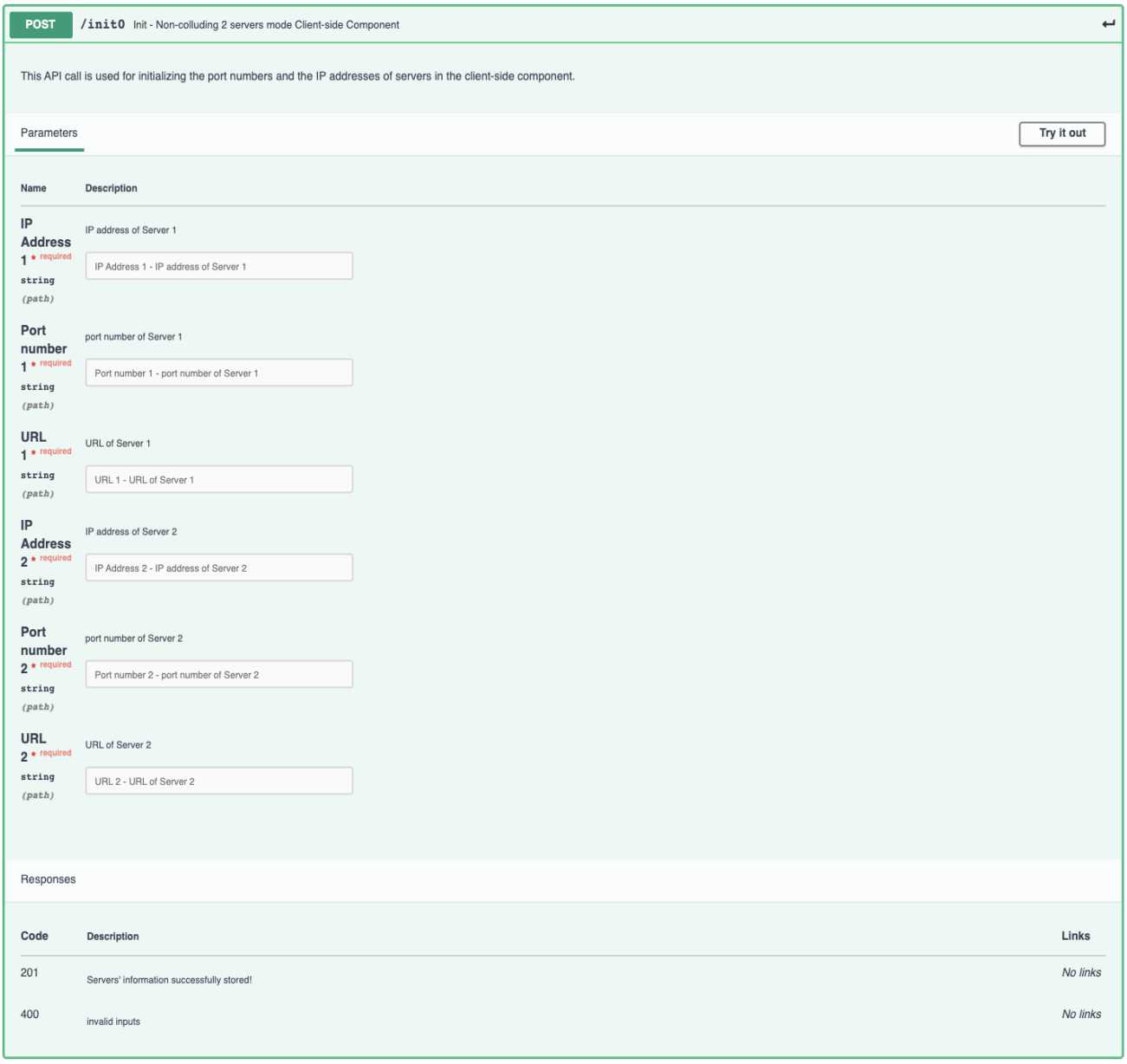
Responses

Code	Description	Links
201	2PC based clustering algorithm has been started!	*No links*
400	invalid inputs	*No links*

Project No. 786767

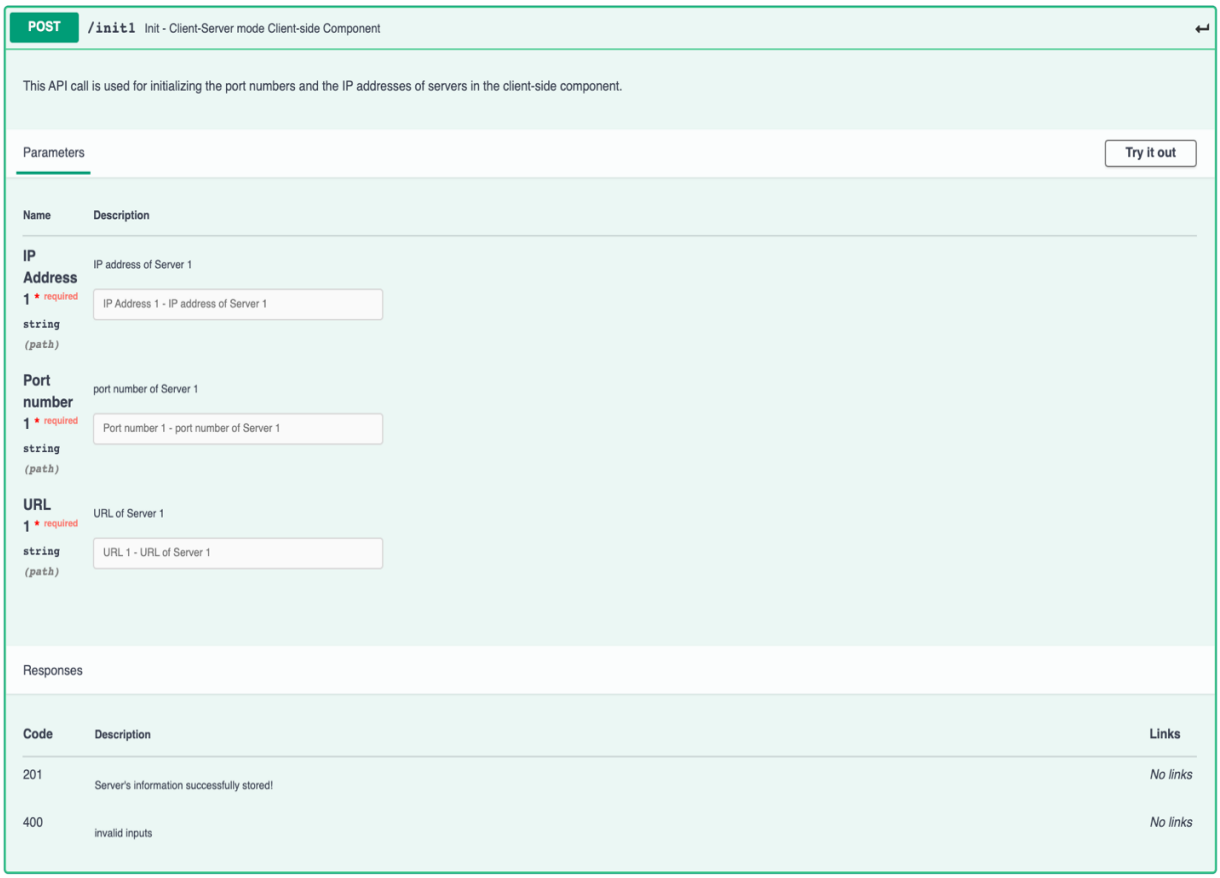
Client-side component API:

Init0



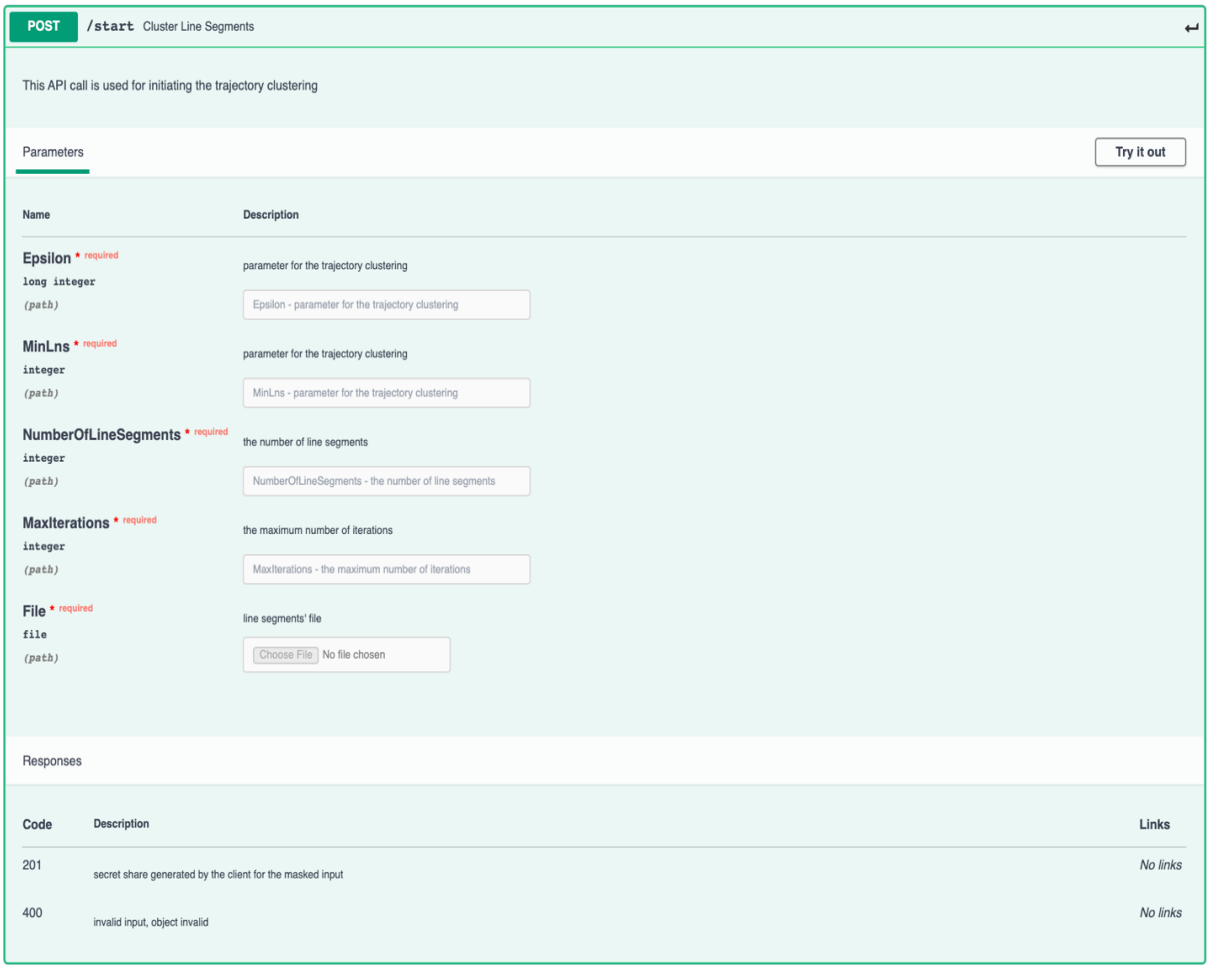
Project No. 786767

Init1



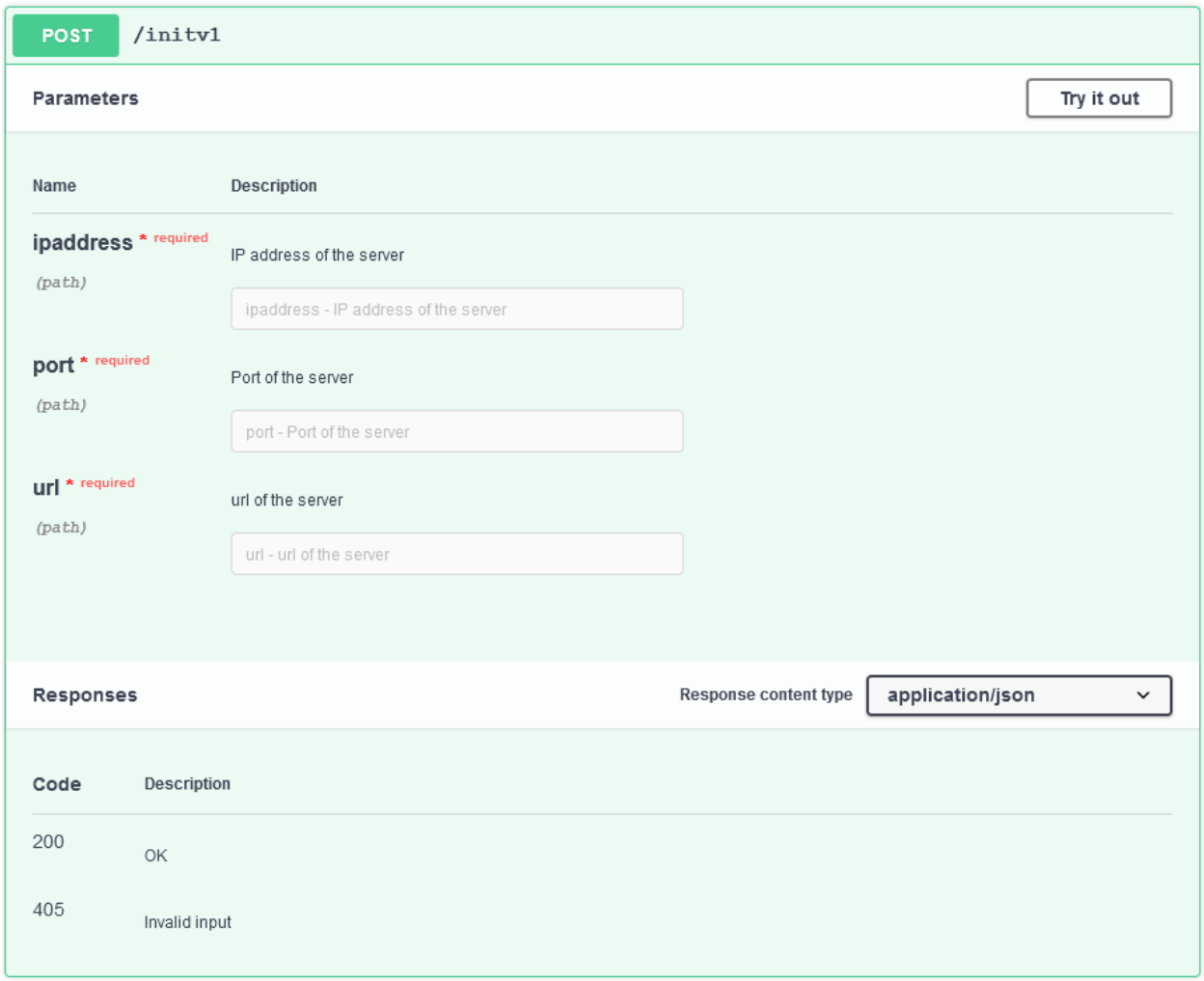
Project No. 786767

Start



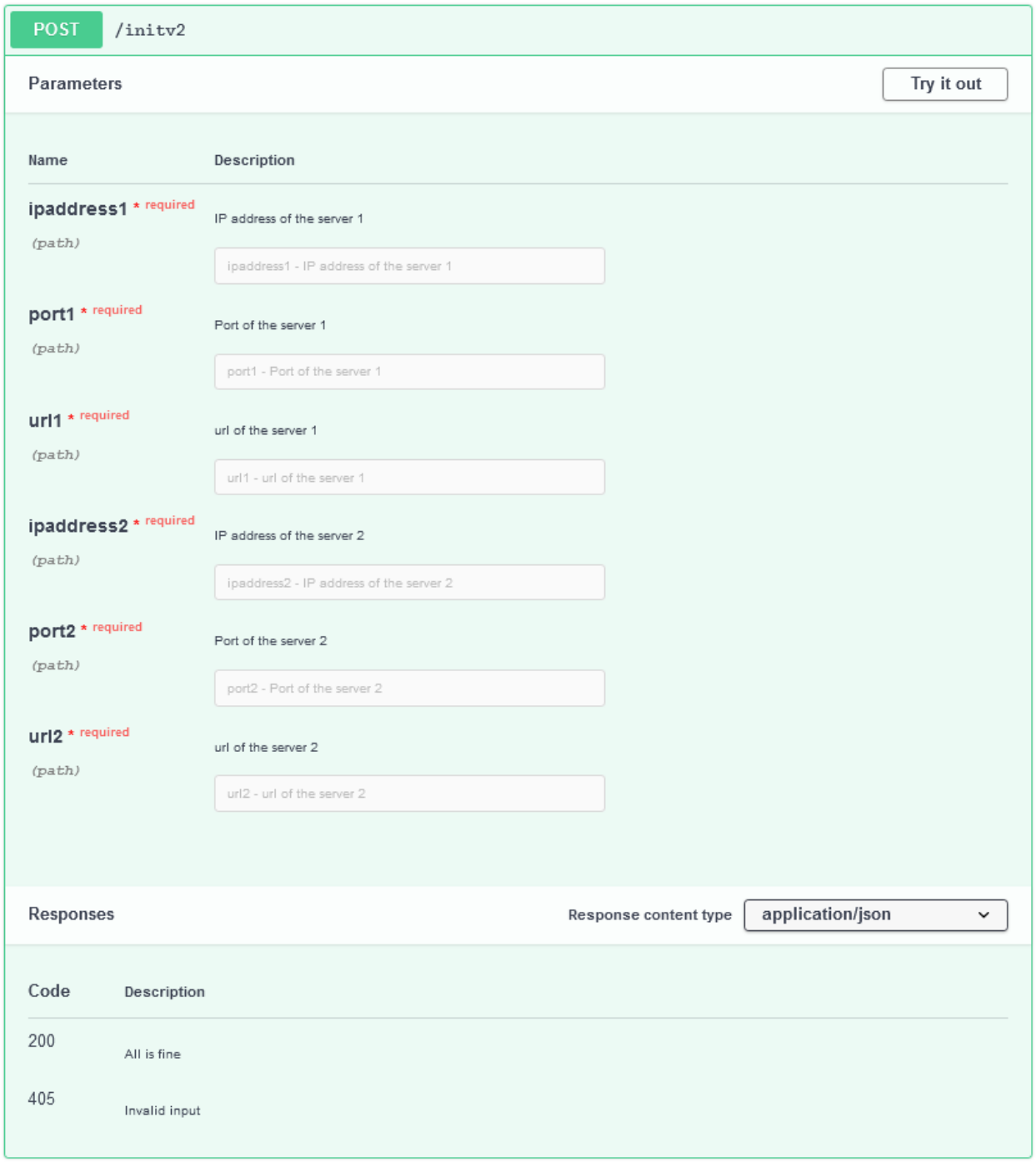
3.2. Privacy-preserving clustering based on MinHash

Initv1



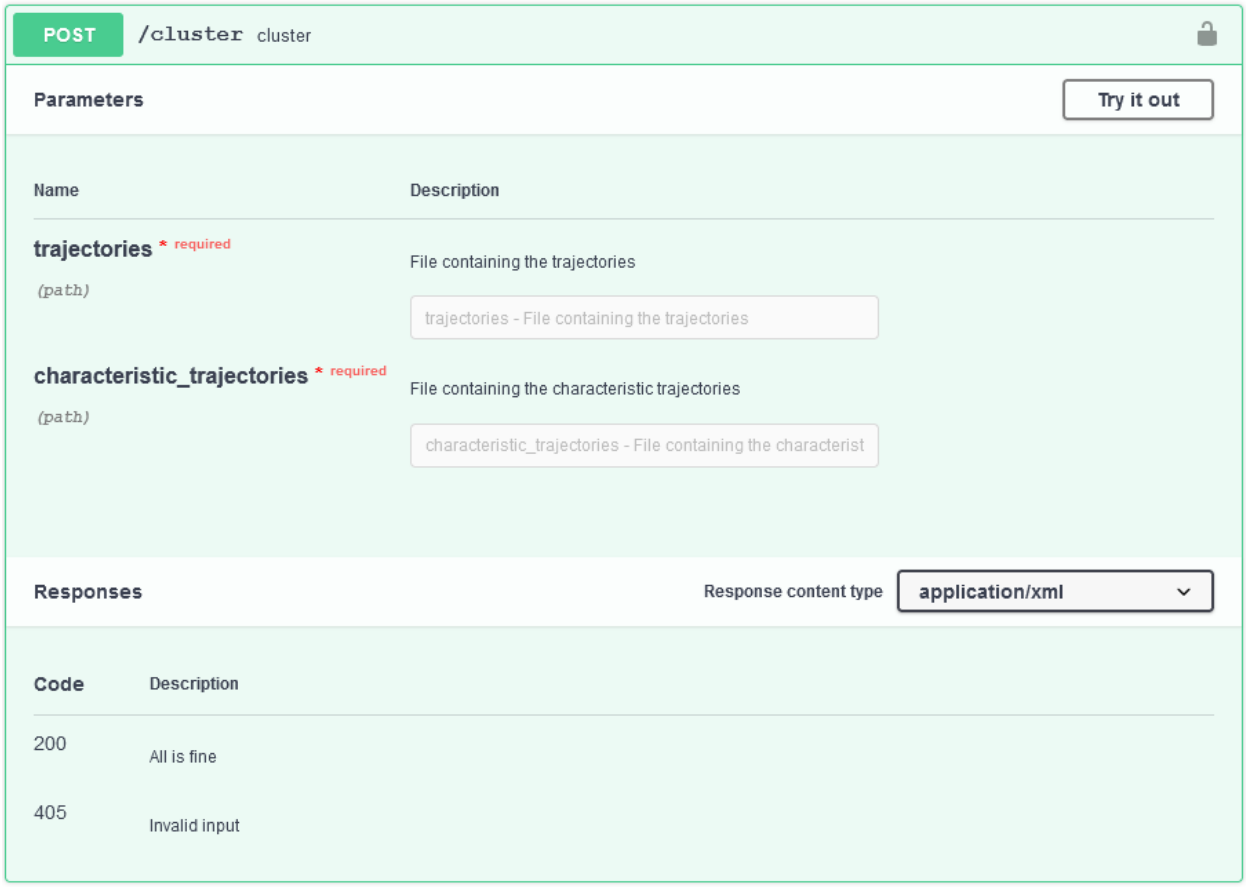
Project No. 786767

Initv2



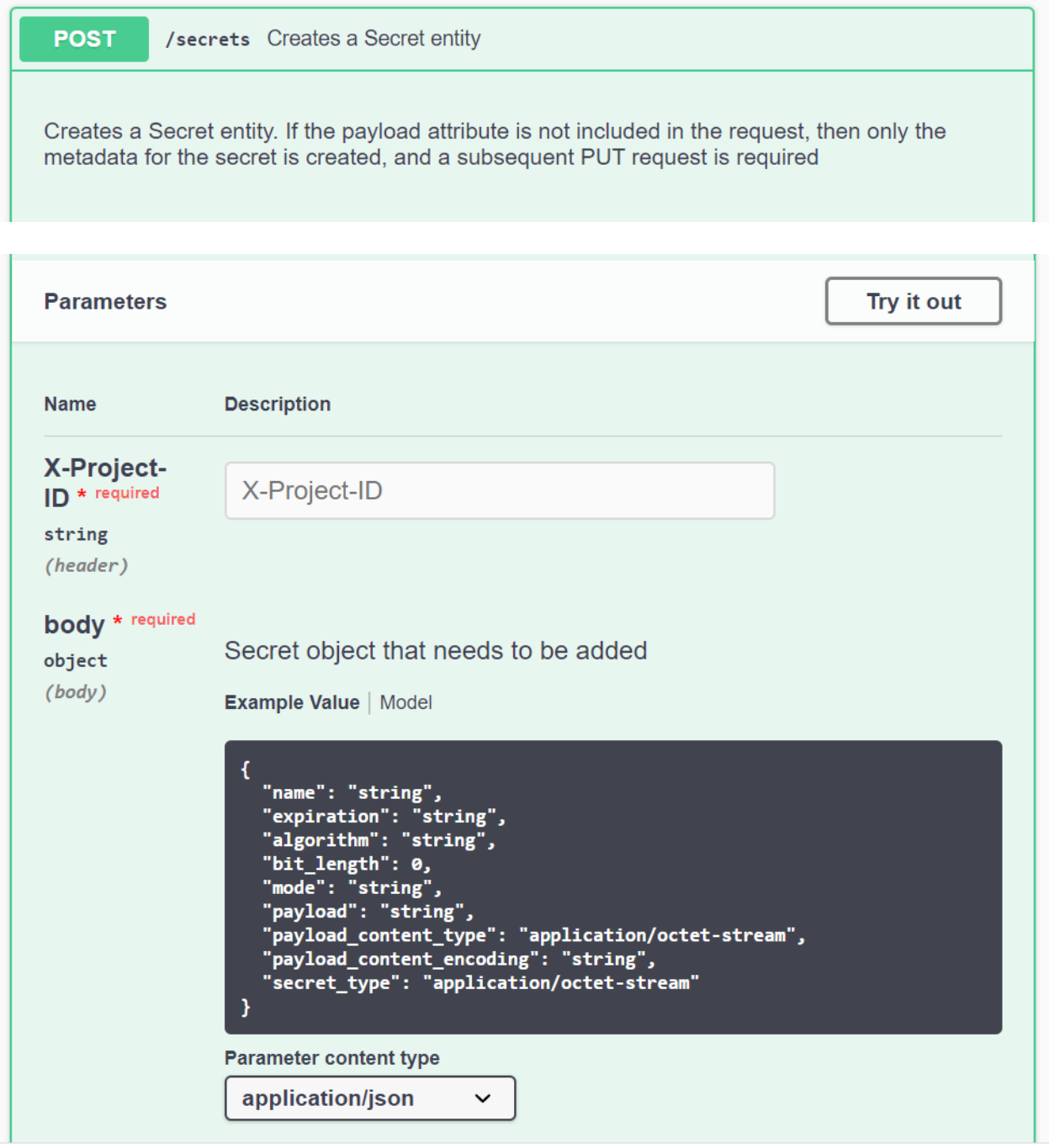
Project No. 786767

Cluster



4. Key Manager

POST/Secrets



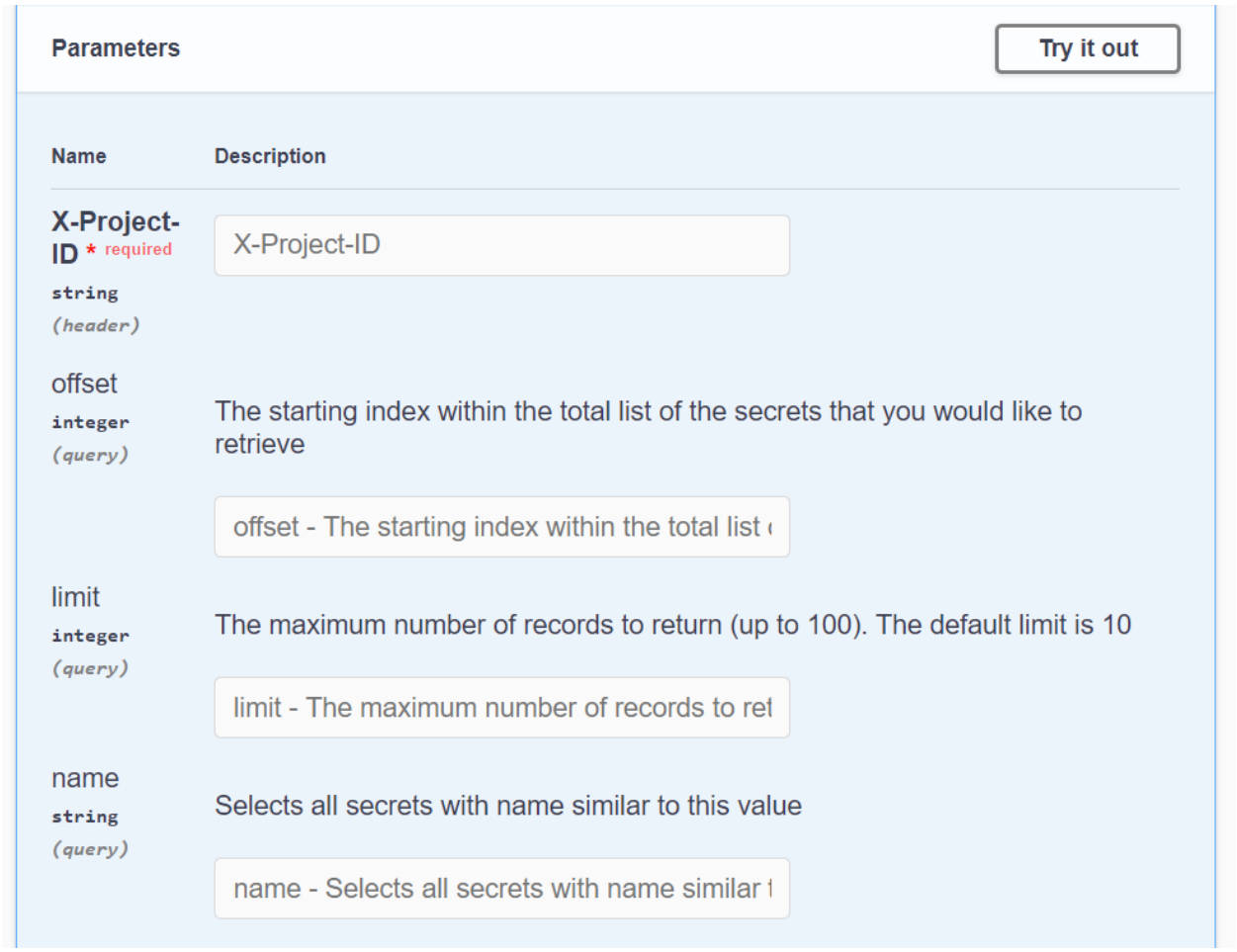
Responses | **Response content type** application/json

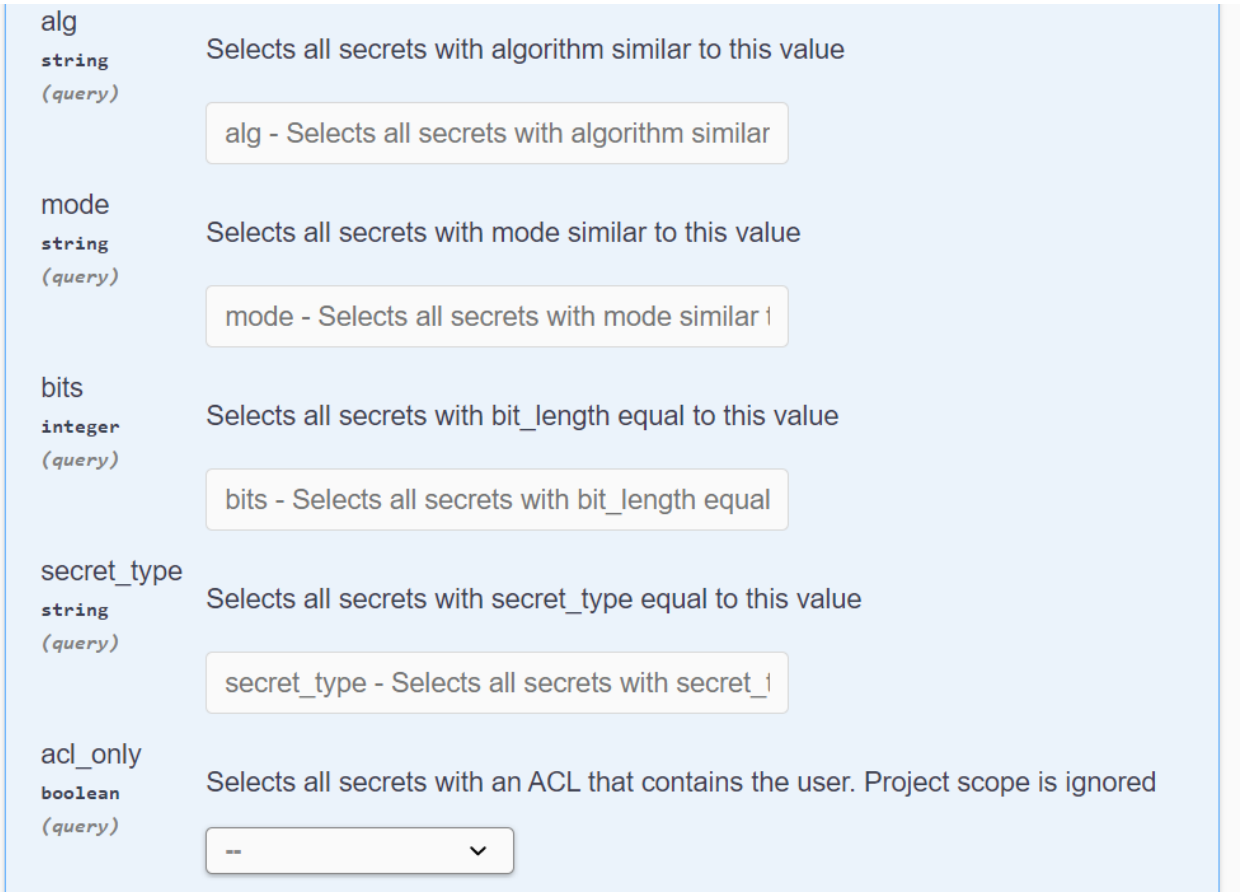
Code	Description
201	Successfully created a Secret **Example Value** | Model `{ "secret_ref": "string" }`
400	Bad Request
401	Invalid X-Auth-Token or the token doesn't have permissions to this resource
403	Forbidden. The user has been authenticated, but is not authorized to create a secret. This can be based on the user's role or the project's quota
415	Unsupported media-type

GET/secrets

GET `/secrets` Lists a project's secrets

Lists a project's secrets. The list of secrets can be filtered by the parameters passed in via the URL. The actual secret payload data will not be listed here. Clients must instead make a separate call to retrieve the secret payload data for each individual secret


Parameters
Try it out
Name
Description
X-Project-ID * required
string
(header)
X-Project-ID
offset
integer
(query)
The starting index within the total list of the secrets that you would like to retrieve
offset - The starting index within the total list
limit
integer
(query)
The maximum number of records to return (up to 100). The default limit is 10
limit - The maximum number of records to ret
name
string
(query)
Selects all secrets with name similar to this value
name - Selects all secrets with name similar


alg
string
(query)
Selects all secrets with algorithm similar to this value
alg - Selects all secrets with algorithm similar
mode
string
(query)
Selects all secrets with mode similar to this value
mode - Selects all secrets with mode similar
bits
integer
(query)
Selects all secrets with bit_length equal to this value
bits - Selects all secrets with bit_length equal
secret_type
string
(query)
Selects all secrets with secret_type equal to this value
secret_type - Selects all secrets with secret_
acl_only
boolean
(query)
Selects all secrets with an ACL that contains the user. Project scope is ignored
--

created
string
(query)

Date filter to select all secrets with created matching the specified criteria. The values for this parameter are comma separated lists of time stamps in ISO 8601 format. The time stamps can be prefixed with any of these comparison operators: gt: (greater-than), gte: (greater-than-or-equal), lt: (less-than), lte: (less-than-or-equal).

created - Date filter to select all secrets with (

updated
string
(query)

Date filter to select all secrets with created matching the specified criteria. The values for this parameter are comma separated lists of time stamps in ISO 8601 format. The time stamps can be prefixed with any of these comparison operators: gt: (greater-than), gte: (greater-than-or-equal), lt: (less-than), lte: (less-than-or-equal).

updated - Date filter to select all secrets with

expiration
string
(query)

Date filter to select all secrets with expiration matching the specified criteria. The values for this parameter are comma separated lists of time stamps in ISO 8601 format. The time stamps can be prefixed with any of these comparison operators: gt: (greater-than), gte: (greater-than-or-equal), lt: (less-than), lte: (less-than-or-equal).

expiration - Date filter to select all secrets wit

sort
string
(query)

Determines the sorted order of the returned list. The value of the sort parameter is a comma-separated list of sort keys. Supported sort keys include created, expiration, mode, name, secret_type, status, and updated. Each sort key may also include a direction. Supported directions are :asc for ascending and :desc for descending. The service will use :asc for every key that does not include a direction.

sort - Determines the sorted order of the retu

Responses Response content type application/json

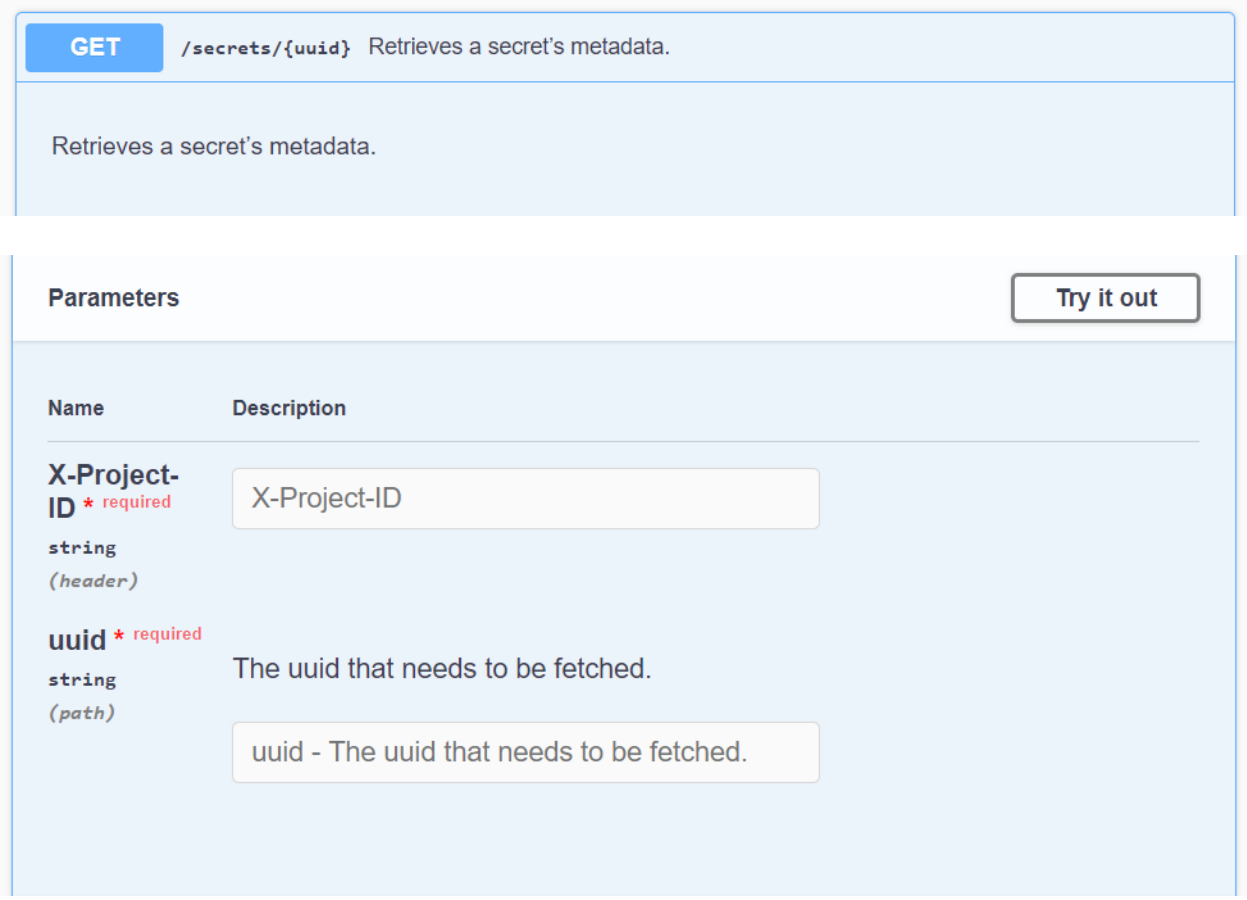
Code	Description
200	Successfull request **Example Value** | Model

```
{
  "secrets": [
    {
      "name": "string",
      "expiration": "string",
      "algorithm": "string",
      "bit_length": 0,
      "mode": "string",
      "payload": "string",
      "payload_content_type": "application/octet-stream",
      "payload_content_encoding": "string",
      "secret_type": "application/octet-stream"
    }
  ],
  "total": 0,
  "next": "string",
  "previous": "string"
}
```

401	Invalid X-Auth-Token or the token doesn't have permissions to this resource

Project No. 786767

GET/secrets/{uuid}



Responses | Response content type: application/json

Code	Description
200	Successful request
401	Invalid X-Auth-Token or the token doesn't have permissions to this resource Example Value | Model


```
{
  "status": "string",
  "created": "string",
  "updated": "string",
  "expiration": "string",
  "algorithm": "string",
  "bit_length": 0,
  "mode": "string",
  "name": "string",
  "secret_ref": "string",
  "secret_type": "application/octet-stream",
  "content_types": "string"
}
```


404	Not Found
406	Not Acceptable

PUT/secrets/{uuid}

PUT `/secrets/{uuid}` Upload payload to a secret

Add the payload to an existing metadata-only secret, such as one made by sending a POST /v1/secrets request that does not include the payload attribute..

Parameters Try it out

Name	Description
X-Project-ID * required `string` *(header)*	X-Project-ID
uuid * required `string` *(path)*	secret that need to be added the payload uuid - secret that need to be added the paylo
body * required `object` *(body)*	Updated secret object **Example Value** | Model

```
{
  "name": "string",
  "expiration": "string",
  "algorithm": "string",
  "bit_length": 0,
  "mode": "string",
  "payload": "string",
  "payload_content_type": "application/octet-stream",
  "payload_content_encoding": "string",
  "secret_type": "application/octet-stream"
}
```

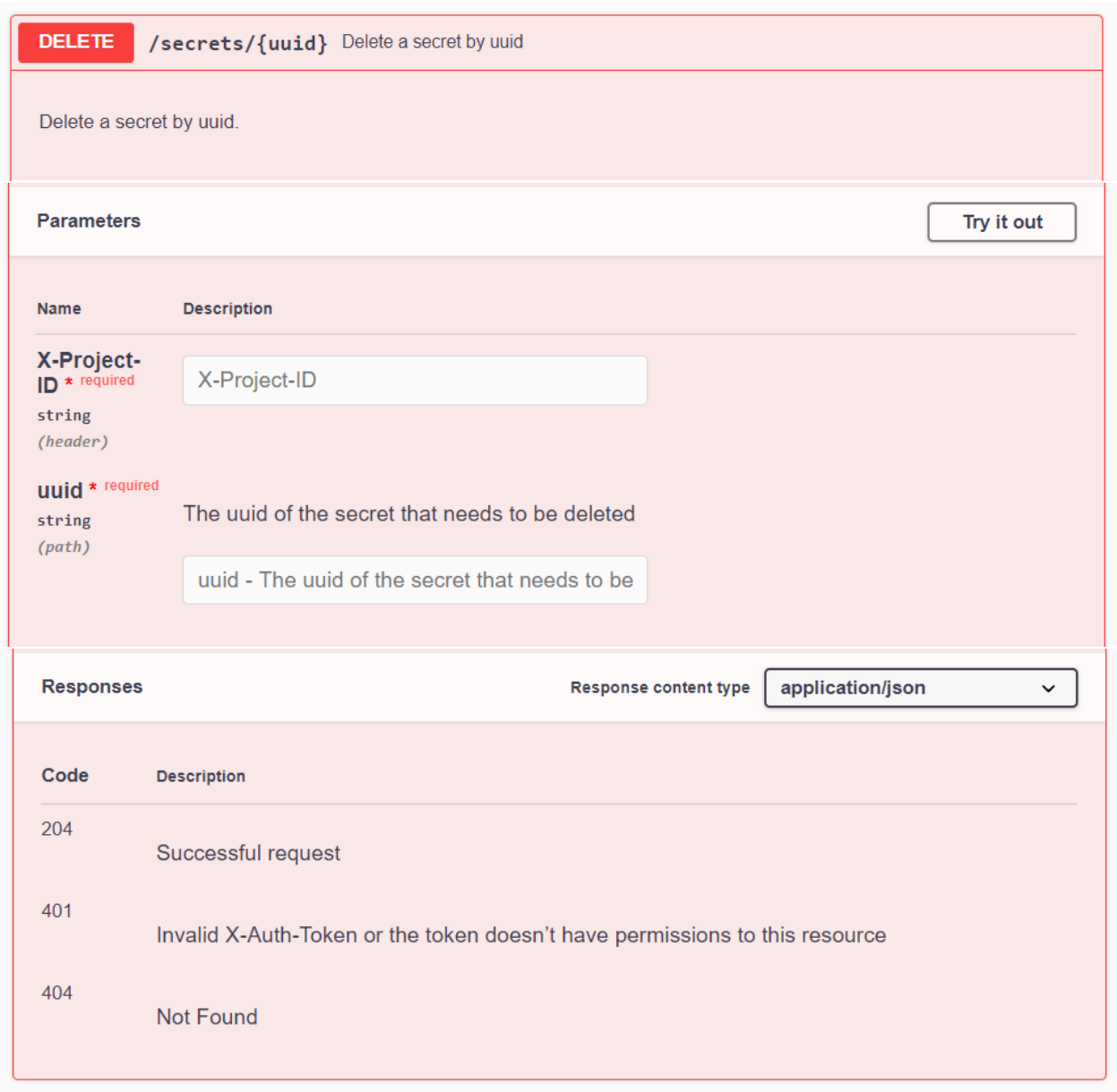
Parameter content type: text/plain

Responses Response content type: text/plain

Code	Description
204	Successful request
401	Invalid X-Auth-Token or the token doesn't have permissions to this resource
404	Not Found

Project No. 786767

DELETE/secrets/{uuid}



Project No. 786767

GET/secrets{uuid}/payload