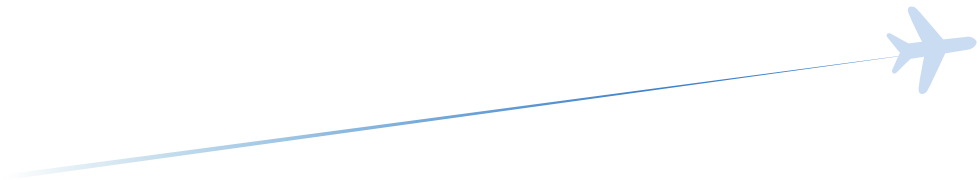

D2.2 System Design Document

Deliverable ID:	**D2.2**
Dissemination Level:	**PU**
Project Acronym:	**SlotMachine**
Grant:	**890456**
Call:	**H2020-SESAR-2019-2**
Topic:	**SESAR-ER4-27-2019 Future ATM Architecture**
Consortium Coordinator:	**Frequentis**
Edition Date:	**23 December 2021**
Edition:	**01.00.01**
Template Edition:	**02.00.03**

Founding Members

EUROPEAN UNION EUROCONTROL

SESAR JOINT UNDERTAKING

SlotMachine

A PRIVACY-PRESERVING MARKETPLACE FOR SLOT MANAGEMENT

This Deliverable is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No 890456 under European Union's Horizon 2020 research and innovation programme.

Abstract

This document provides the system design related to SESAR ER project SlotMachine. In particular, this deliverable describes the architecture and main components of the SlotMachine system as well as the interfaces for the components and the interaction between components. The SlotMachine system is based on a service-oriented architecture, each component running in its own container and providing a REST interface. The system design proposed in this document serves as the basis for the implementation but will be subject to change during the development process. Some details require further research and experimentation before a final decision regarding the design options can be made. It is recommended to read this deliverable after D2.3 – Business Concepts and D2.1 – Requirements Specification.

Table of Contents

List of Figures

1 Introduction

1.1 Purpose of the document

The purpose of this document is to derive a system design from the requirements specified in D2.1 – Requirements Specification.

This document covers the overall architecture and detailed description of functional components and their interworking. Additionally, this document describes how to validate the proposed design through end-to-end tests based on scenarios from D2.1 – Requirements Specification.

The System Design addresses the "how" of SlotMachine and proposes various implementation options that are used for subsequent evaluation. The proposed design, however, may be adapted in the course of the agile development method during demonstrator implementation. Furthermore, some design choices are still subject to change based on experimentation when the components have been implemented as a first version.

1.2 Intended readership

The target group for this document is the SlotMachine project team with a special focus on R&D related topics. This document also provides input for the Advisory Board to describe detailed specification assets, and, beyond the project, product managers and system architects can use the content of this document as a basis for implementation and further development.

1.3 Background

This document took references from the TS (Technical Specification) document structure and provides a comprehensive overview about design and verification of the planned solution.

1.4 Structure of the document and relation to other deliverables

The general structure of this document is as follows.

- Chapter 1 (this section): It provides a general idea of the entire document. It includes the purpose, readership, inputs from other projects, component purpose and high-level overview and acronyms used in the document,
- Chapter 2 provides an overview of the SlotMachine system at a conceptual level,
- Chapter 3 gives describes the SlotMachine system's main components,
- Chapter 4 specifies the interfaces of the components,
- Chapter 5 provides sequence diagrams that explain the interactions between the components,
- Chapter 6 documents the approach for implementation and verifying project progress,
- Chapter 7 concludes this document,
- Chapter 8 lists all referenced documents,
- Appendix A lists and defines selected terms used in the document.

This document relates to the other deliverables of the SlotMachine project as follows.

- D2.1 – Requirements Specification [1]: The requirements are the foundation for the proposed design in this document; the design refers to the requirements in D2.1.
- D2.3 – Business Concepts [2]: More details on operational background, deployment options, and market mechanisms can be found in D3.2.
- D3.2 – Specification of the PrivacyEngine component [3]: The Privacy Engine and related components (MPC Nodes and Credit Handling) are described in further detail in D3.2.
- D4.2 – Specification of the Evolutionary Algorithm [4]: The Heuristic Optimizer is described in further detail in D4.2.
- D5.1 – SlotMachine Platform Demonstrator [5]: The platform demonstrator will integrate the individual components for evaluation.

2 Overview

In case of reduced capacity of the air traffic network, the Network Manager (NM) initiates a regulation, which typically causes flight delay. In case of a regulation, the conventional approach to re-planning the departure times of flights is to follow the principle of "first-planned, first-served". For airspace users, however, different flights may have different priorities due to the individual cost structures of different flights. The SlotMachine system will allow airspace users to submit preferences regarding the departure times of individual flights, which are then considered during an optimization session that aims to find a globally optimal flight list. We refer to D2.3 – Business Concepts [2] for more information on the operational and economic background.

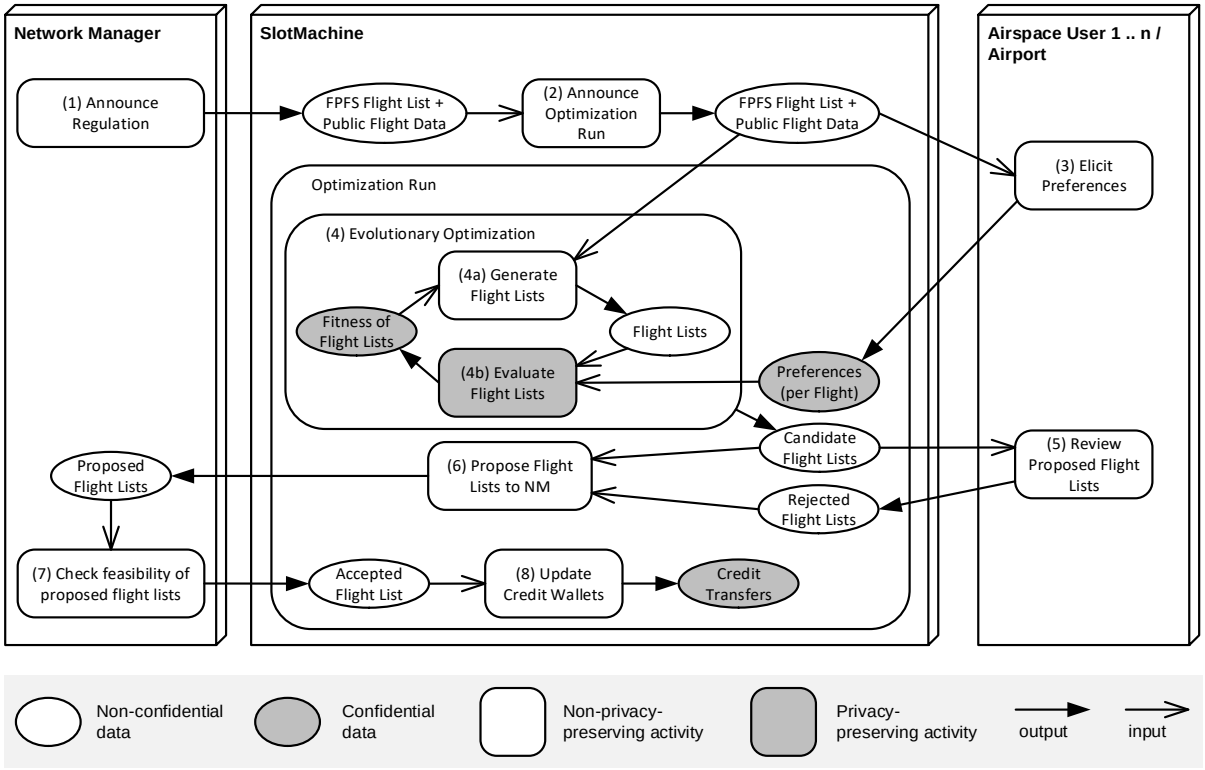

Figure 1. Conceptual representation of the flow of a SlotMachine optimization run

From a conceptual perspective, the flow of a SlotMachine optimization run is as follows (Figure 1). Once the network manager announces a regulation, the SlotMachine system retrieves the new first-planned first-served (FPFS) flight list and any non-confidential data related to the flights affected by the regulation. The SlotMachine system then initiates an optimization run for the regulation. The SlotMachine system may conduct multiple optimization runs per regulation, so the optimization run may only involve a subset of the flights affected by the regulation. The airspace users (and possibly the airport) are notified about the regulation and receive a flight list for the time segment that the optimization run covers. The airspace users (and possibly the airport) then elicit preferences regarding the prioritization of the flights that are affected by the regulation. The preferences are confidential and therefore submitted in encrypted form; the activities processing the preferences are privacy-preserving. Following the submission of the preferences by the airspace users, the SlotMachine system conducts an evolutionary optimization run to find optimized candidate flight lists. The optimization run is an evolutionary process that consists of finding a population of flight lists and evaluating the flight

lists. Evaluation of the flight lists in a population of flight lists is a privacy-preserving activity, employing (secure) multiparty computation to compute fitness values for the proposed flight lists. After the evolutionary optimization run has finished, the best candidate flight lists are submitted to the airspace users and the airport for review. The airspace users and airport may reject individual solutions. The acceptable candidate flight lists are proposed to the Network Manager. The Network Manager checks the proposed flight lists' feasibility, chooses a flight list, and communicates the accepted flight list back to the SlotMachine system, which relays the accepted flight list back to airspace users and airport before updating the credit wallets if a credit-based market mechanism is used (see D2.3 – Business Concepts [2]).

2.1 Component Overview

Figure 2 gives an overview of the SlotMachine system's main components. The individual components will be implemented as services that run in separate containers. Microservice design is used for best portability, allowing for the components to be easily run in cloud environments.

The high-level data flow through the different components is as follows. Note that due to space considerations, for the sake of comprehensibility, not all the proposed interfaces and components are represented in Figure 2. We refer to Chapter 3 for more details on the components, Chapter 4 for a more detailed description of the interfaces, and Chapter 5 for sequence diagrams of the interaction between the components.

1. The main component is the Controller, which manages the optimization run. The Controller regularly checks with the Network Management Function (NMF) which regulations have been issued and, if a regulation has indeed been issued, retrieves the list of flights affected by the regulation. The NMF is the component that realizes the functionality that is currently assumed by the Network Manager.

2. The Controller then notifies the airspace users and the airport of a regulation.

3. The Airspace User component consists, on the one hand, of a web-based user interface (AU web app) that allows representatives to login and submit preferences regarding an optimization run. On the other hand, the Airspace User component consists of a REST interface for interaction with the Controller (AU REST). The Controller submits notifications of regulations only to AU components that registered for notifications of a certain regulation type (ARRIVAL, DEPARTURE or EN ROUTE) on a specific airport (not shown in Figure 2). Note that an operative system would also have to provide an Airport component, the development of which will be derived from the AU component.

4. After receiving the notification about a regulation, the AU web app loads the flight list for the airspace user and displays the flights in a list in the user interface (UI). The UI also shows the cut-off time, i.e., the time when the next optimization run is scheduled; up to the cut-off time the AU is able to push preferences regarding the optimization of the flight list.

5. The airspace user representative can enter for each flight a time wished, a time not before, and a time not after, which are also referred to as margins, as well as a priority.

6. The AU REST component translates the margins and the priority into a set of weights/utilities (see also D2.3 – Business Concepts [2]) following a configurable, airline-specific ruleset. The weights are encrypted for privacy reasons and forwarded to the Controller.

7. After reaching the cut-off time, the Controller forwards the preferences to the Privacy Engine where the solutions to the flight prioritization problem (optimized flight lists) are evaluated.

8. The Heuristic Optimizer component looks for better solutions to the flight prioritization problem in an iterative way, submitting the solutions found in each iteration step to the Privacy Engine for evaluation.

9. The Privacy Engine does not decrypt the submitted preferences of the airspace users but employs multiple MPC nodes, which are run independently from the SlotMachine system, e.g., by airspace users themselves, to evaluate the flight lists found by the Heuristic Optimizer.

10. When the optimization run has concluded, the Controller forwards a set of optimized flight lists to the to the Airspace User component. The solutions are presented on the AU web app for evaluation.

11. The airspace user has the possibility to reject one or more solutions.

12. The acceptable flight lists, i.e., those that are not rejected by anyone, are submitted to the NMF, which takes the decision about which of the proposed solutions is accepted.

13. The accepted solution will be communicated to the Controller by the NMF. The Controller notifies the AU component (and the Airport component) of the new flight list.

14. The Controller, together with a Credit Management component and the Privacy Engine updates the credit wallets. Transactions are also recorded in a blockchain.

In this project, we implement the different components as services. A service-oriented architecture is common-place in systems development, particularly in air traffic management (cf. System Wide Information Management). Most components will be implemented using a microservice architecture based on the RESTful web service paradigm [6]. The advantage of such an architecture is that it can be easily run in a cloud environment and allows for horizontal scaling through parallelization and distributed computation (see requirements port_1–3 and rel_1 in D2.1 [1]).

D2.3 – Business Concepts [2] proposes different market mechanisms. The mechanism SM1 is based on utilities expressed in real-world currency, e.g., euros. In SM1, no credit management is necessary but other clearing options would have to be investigated. The mechanism SM3 replaces real-world currency with credits. The mechanism SM2 is based on combinatorial auctions and also employs credits. In both cases, however, different interfaces would be required.

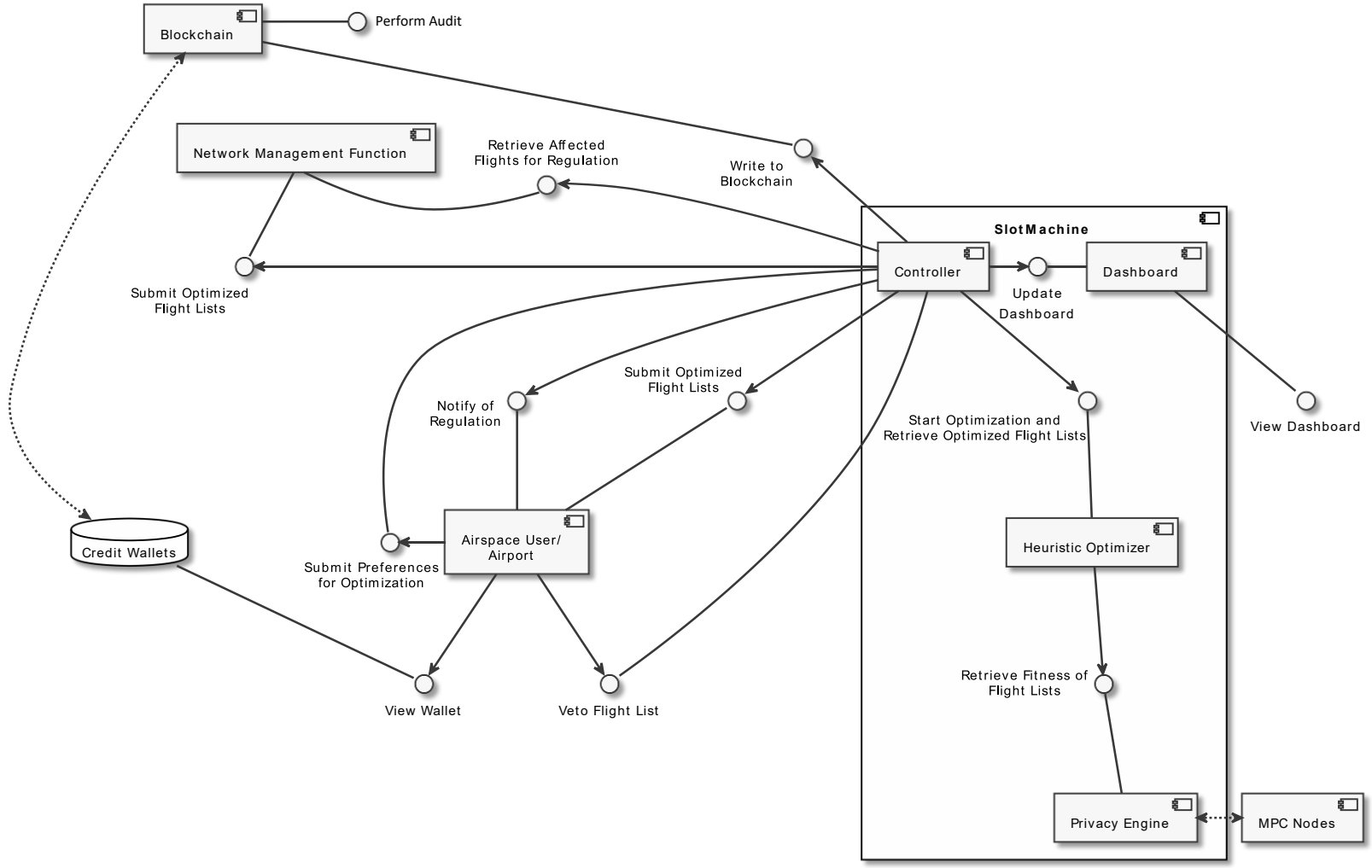

Figure 2. Main components and interfaces involved in the flight prioritization process

2.2 Demonstrator Iterations

During the lifetime of the project a demonstrator to demonstrate, evaluate, and compare certain functions will be developed according to D2.3 Implementation Use Case" using an iterative approach, which means functionalities and components will be added step by step. The identified requirements will guide the implementation of different demonstrators which are developed iteratively. Ideally the final iteration should satisfy all identified requirements. However, part of the project is to find out whether the requirements can be satisfied by a SlotMachine implementation or whether they are even necessary. In particular, we will develop the following iterations of the demonstrator:

1. The first "**non-privacy-preserving**" version (planned for October 2021) of the demonstrator will cover initial demonstration of submitting preferences by AUs (via the AUs web app through the Controller to the Optimizer) and optimizing a flight list to discuss first results. This demonstrator involves basic implementation of following components:
    - AU web app and AU REST
    - Controller
    - Network Management Function (NMF)
    - Heuristic Optimizer
    - Dashboard

    A variant of the non-privacy-preserving demonstrator will use the Privacy Engine component and MPC Nodes using a non-privacy-preserving configuration of those components.

2. In the 2nd iteration (planned for February 2022) of the demonstrator, **"privacy-preserving"** mechanisms will be added through the Privacy Engine (incl. MPC nodes) and end-to-end encryption between AU clients and MPC nodes; a Dashboard provides an overview and aggregated insights over the optimizations performed by the components. This demonstrator involves basic implementation of following components:
    - AU web app and AU REST
    - Controller
    - NMF
    - Heuristic Optimizer
    - Privacy Engine
    - MPC Nodes
    - Dashboard

3. In the 3rd version (planned for May 2022), **secure, and traceable credit handling** will be provided through blockchain technology to allow greater flexibility and more options for airlines participating in the marketplace. In this state all mentioned components will be involved.

4. Afterwards the demonstrator is iteratively improved based on feedback from Advisory Boards (2nd Advisory Board in March 2022, 3rd Advisory Board in Summer 2022) and evaluating various datasets (good cases, bad cases, real-world cases) and deployment scenarios (as described in D2.3 – Business Concepts, Section 4.5).

3 Functional Components

In the following, we briefly describe the proposed components for the SlotMachine system. We refer to Chapter 5 for more detailed information on the information flow between the components. We link the components to the relevant requirements from D2.1 – Requirements Specification [1]. An evaluation of the fulfilment of the requirements will be part of D5.1 – SlotMachine Platform Demonstrator [5]. We refer to D3.2 – Specification of the Privacy Engine Component [3] for more details on the Privacy Engine and the MPC nodes as well as credit management. We refer to D4.2 – Specification of Evolutionary Algorithm [4] for more details on the Heuristic Optimizer.

3.1 Airspace User (AU)

Relevant requirements: loc_1–2, usab_1–6, usab_7, priv_7, sm_1, au_1–7

The Airspace User (AU) component consists of two subcomponents. The AU web application (webapp) and the AU REST Endpoint. The webapp provides basic functionality for end users (AU representatives) to manage participation in the SlotMachine optimization run, i.e., functionality for login, registration for regulation for a specific airport, getting the current flight list for a regulation, submitting AU preferences of slots for flights in form of margins and priorities, and informing AUs about upcoming optimization runs, and the confirmed optimized flight list after the optimization run. For those interactions with the SlotMachine system, the AU webapp uses the functionalities the AU webapp uses a REST endpoint (AU REST). This AU REST component could also be used by AUs to plug-in their own modules that automatically derive and submit slot preferences per flight in the future.

The user interface (UI) design and implementation will be conducted as an iterative process, in close collaboration with the representatives of SWISS, whose knowledge of the preferences of prospective users and insights into the current processes regarding flight prioritization will help shape the appearance of the UI.

Which definition of “slot” is taken into consideration for slot optimization depends on the type of the currently active regulation.

- For *departure* regulations a slot is a flight’s calculated take-off time (CTOT).
- For *arrival* regulations the slot is the flight’s CTOT at the departure airport plus the estimated elapsed time (EET) on arrival at the destination airport, i.e., the arrival time at destination.
- For *en route* regulations the relevant time slot for the respective regulation is calculated based on the estimated elapsed time (EET) added to the previous way point of the flight; en route regulations are outside the scope of this project.

3.2 Network Management Function (NMF)

Relevant requirements: port_1-3, rel_1-4, priv_15, co_2-3, nm_1–2

The Network Management Function component simulates EUROCONTROL’s Network Manager (see [7]) within the SlotMachine project and is a service for providing initial slot configurations. The NMF also accepts or denies the slot rearrangements found by the SlotMachine optimization run, publishing

the finally accepted solution to a flight prioritization problem. The NMF component interacts with the SlotMachine's Controller component (see Section 3.3) and implements the Network Manager B2B interface to provide realistic test data (see Section 6.1 for further information on the test setup).

The data format for the initial slot configuration is based on the FlightListByAerodrome request and reply format as specified in the NM B2B interface (Edition No. 25.0.0.7.142 from 08/09/2021), which is described in Figure 3 and Figure 4, respectively.

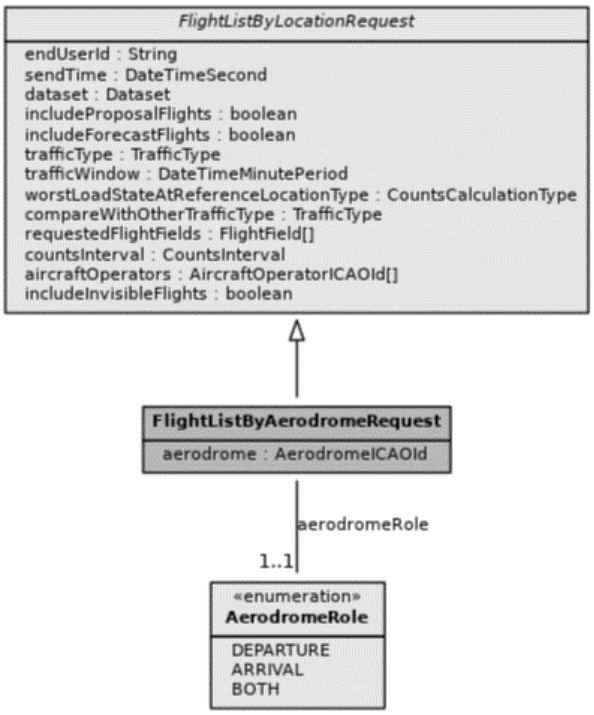

Figure 3. FligthListByAerodromeRequest class diagram [7]

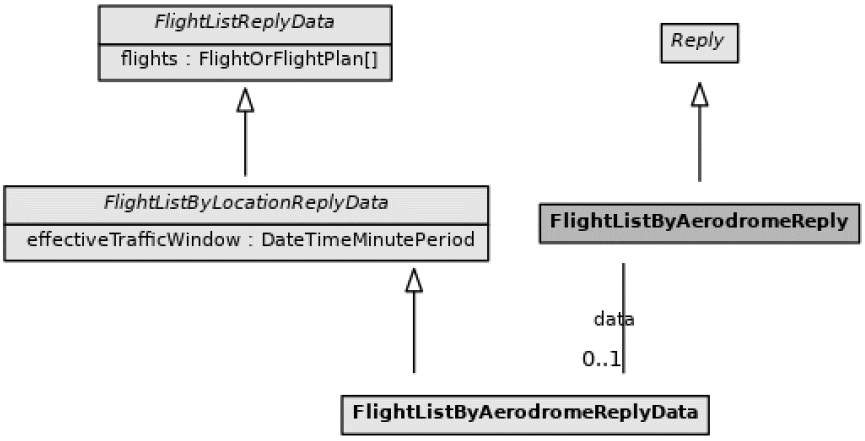

Figure 4. FligthListByAerodromeReply class diagram [7]

3.3 Controller

Relevant requirements: co_1–8, ho_2–4, ho_8, au_2, fair_4, sm_1–4

The Controller is the central component of the SlotMachine system. The Controller handles the SlotMachine system's communication with the AU/Airport components as well as the NMF component. First, the Controller relays information regarding the regulations from the NMF to the AU/Airport component instances. The AU/Airport instances register with the Controller for updates regarding newly issued regulations of a certain kind (departure, arrival, en route) for a particular airport and may then obtain the flight list issued by the NMF, including additional information about the flights, as well as information about the next optimization run to be conducted by the SlotMachine system. Second, the Controller collects the airspace users' slot preferences for the different flights and initiates an optimization run. In order to be able to conduct an optimization run in a privacy-preserving manner, the Controller also configures the Privacy Engine for use by the Heuristic Optimizer (see Section 3.4) during an optimization run.

When initiating an optimization run, the Controller relays the AU/Airport preferences to the Heuristic Optimizer. After the Heuristic Optimizer has finished running an optimization, the Controller collects the result of the optimization run from the Heuristic Optimizer. The result of an optimization run – a ranked list of optimized flight lists – are the candidates for replacing the current flight list and are communicated to the AUs and the Airport. All candidate flight lists that are not vetoed by AUs and Airport are acceptable solutions; the Controller collects the vetoes. After a specified time, the Controller submits the ranked acceptable, i.e., not vetoed, candidate flight lists to the NMF for confirmation. The NMF returns the finally accepted flight list to the Controller, which communicates the new flight list to participating AUs and Airport. Once a new flight list has been accepted by the NMF, the Controller takes care of updating the credit wallets via the Credit Management component (see Section 3.10).

3.4 Heuristic Optimizer

Relevant requirements: ho_1–9, perf_1–4, perf_6

The Heuristic Optimizer is the component that searches for better flight lists with respect to the initial flight list given the preferences submitted to the SlotMachine system by the airspace users. The Heuristic Optimizer employs an evolutionary algorithm to find solutions to the flight prioritization problem. The prime candidate type of evolutionary algorithm considered for the implementation of the Heuristic Optimizer is the genetic algorithm (see D4.1 – Report on State of the Art of Relevant Concepts [8]). We refer to D4.2 – Specification of Evolutionary Algorithm [4] for more details on the implementation of the evolutionary algorithm.

Intuitively, the Heuristic Optimizer generates a "population" of candidate solutions to the flight prioritization problem, i.e., a set of flight lists, in each iteration step. The Heuristic Optimizer then submits the population of candidate solutions to the Privacy Engine for evaluation (see Section 3.7). Each flight list can be evaluated using a fitness function that returns a fitness value that allows to judge how good a solution is with respect to the preferences submitted by the airspace users. The Privacy Engine ranks the solutions in the population by fitness value and returns a maximum (and possibly a minimum fitness value) for the population. The Privacy Engine, ideally, does not return absolute fitness values for each flight list because that would allow a curious platform provider to log the fitness values and infer the airspace users' preferences.

The design of the Heuristic Optimizer aims at allowing to find an acceptable solution to the flight prioritization problem within the given time window. Hence, one guideline for the development of the Heuristic Optimizer will be the support for parallelization as well as the possibility to abort optimization runs at any time while still being able to obtain valid results from aborted optimization runs. Implementation as genetic algorithm allows for such an implementation.

3.5 Airport

Relevant requirements: loc_1–2, usab_1–6, usab_7, priv_7, sm_1, au_1–7

The implementation of the Airport component is outside the scope of the project and thus not planned for the SlotMachine demonstrator. In an actually deployed SlotMachine system, however, airports should have an interface as well in order for staying up-to-date regarding the optimization runs, possibly even submitting preferences for flight prioritizations themselves. The Airport component would be derived from the AU component, following the same general architecture with a REST interface for interaction with the Controller on the one hand and a user interface or expert system on the other hand for eliciting the preferences regarding flight prioritization.

3.6 Dashboard

Relevant requirements: db_1

The Dashboard component provides public aggregated information about evaluated and confirmed optimizations as well as spent and earned credits. The Dashboard component consists of a public dashboard and an airline-specific private dashboard. The public dashboard does not provide any detailed airline-specific data. The private dashboard, on the other hand, is a secured/protected area for airlines to show AU-specific details. In case of the public dashboard, data for aggregation and display are merely collected from the Controller component. The data for the private dashboard will have to be collected by the airspace users themselves.

During the implementation and testing phase of the SlotMachine project, another area of the Dashboard is the development area where developers can see details about optimization runs and results. The Dashboard component itself will be implemented/configured using an iterative approach together with representatives from SWISS.

On the public dashboard, the following data will be displayed in order to give an overview of the activity on the SlotMachine platform.

- Average number of swapped flights per optimization run
- Total number of credits earned through optimization runs
- Total number of optimization runs
- Ratio between rejected solutions and found solutions by optimization runs
- Average number of flights per optimization run

The technical realization of the private dashboard still needs to be investigated. A particular challenge is to keep the private information also hidden from the SlotMachine system to improve stakeholder

acceptance. For the private dashboard, the following detailed slot swapping information will be displayed (subject to revision if during the Advisory Board and UI design sessions with SWISS different preferences emerge).

- Airspace User: the user that maintained the flight optimization
- Aerodrome of Departure
- Aerodrome of Destination
- The type of the regulation the flight belongs to
- The unique identifier of the flight
- The estimated flight time (estimated elapsed time, EET)
- The initial estimated take-off time
- The priority set by the airspace user
- The time not before set by the airspace user
- The time not after set by the airspace user
- The time wished set by the airspace user
- The credits spend/earned for a specific flight swap
- The initial CTOT before the swap
- The new CTOT (“slot”) as outcome of the optimization run

Figure 5 gives a conceptual overview of the data flow to the dashboard. A metric data collector receives metric data from multiple providers. The collected metric data are the input for generating visualizations, which are then displayed in the dashboard.

Figure 5. Overview of the data flow to the dashboard

3.7 Privacy Engine

Relevant requirements: pe_1–17, rel_7, priv_8, ho_6–7, mpc_3, mpc_5

The Privacy Engine (PE) component assists the Heuristic Optimizer to find optimal solutions for the flight prioritization problem. In particular, the PE is responsible for the generation of rankings for suggested solution to the flight prioritization problem based on the private inputs submitted by the AUs. The PE leverages multiparty computation (MPC) to conduct the ranking of the solutions in a privacy-preserving manner, i.e., by computing over the AUs' encrypted input data, which are thereby kept confidential. The basic technologies used in the PE have been described in *D3.1* [9] and a detailed specification of the component is given in *D3.1* [3].

The PE component is central to the management of the involvement of the MPC nodes (see Section 3.8) and for the computation of the privacy-preserving rankings as part of the optimization runs. To this end, the PE exposes an easy-to-use REST interface for the Heuristic Optimizer but prevents the SlotMachine from access to any sensitive information. A high-level overview of the interaction between Controller, Heuristic Optimizer, PE, and MPC nodes is shown in Figure 6. The AUs submit encrypted preferences to the Controller of the SlotMachine system, which forwards the encrypted preferences to the Privacy Engine. The Heuristic Optimizer finds new solutions, which are sent to the PE for evaluation. Based on the evaluation results returned by the PE, the Heuristic Optimizer revises the solutions (flight lists). The optimization run is an iterative process. The PE employs MPC nodes, which are outside of the SlotMachine system, to perform the privacy-preserving evaluation of the flight lists submitted by the Heuristic Optimizer.

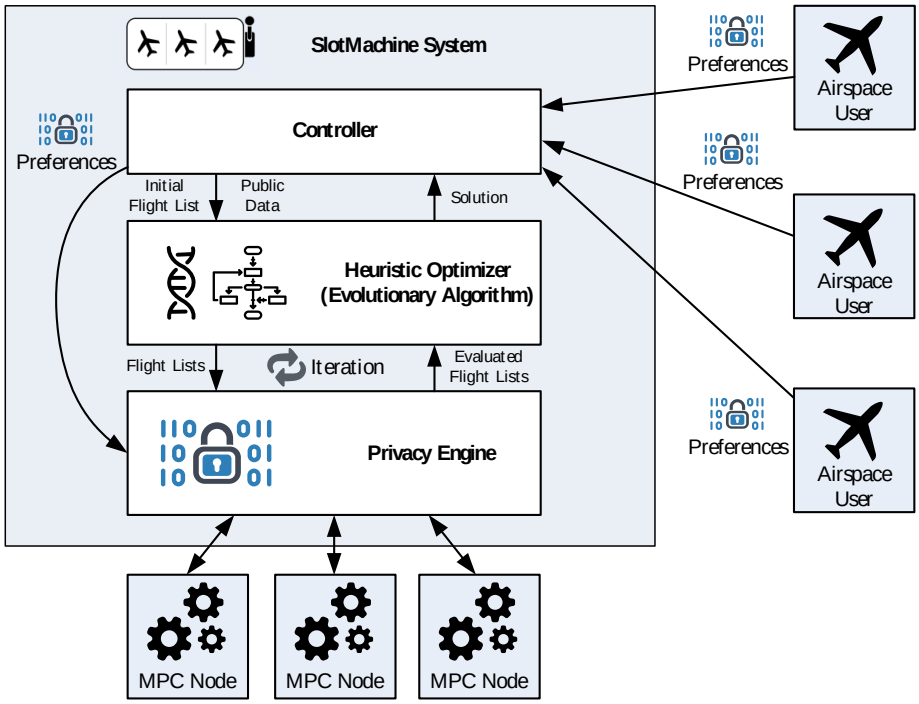

Figure 6. Interaction between Privacy Engine and Heuristic Optimizer as well as MPC nodes

In addition to the core evaluation service offered within PE, a dedicated encoding service is also provided. It is responsible for the client-side encoding of the AU private inputs. The encoding service can be either used as a service or also run locally by AUs.

3.8 MPC Nodes

Relevant requirements: mpc_1–17, pe_10–11

To achieve highest security standards in SlotMachine we use cryptographic techniques for both, to protect business secrets and to enable transparency in the system. To protect business secrets of airlines which are necessary for the platform to operate and optimize slot usage we are using multiparty computation (MPC) among others. This enables us to keep all sensitive information encrypted in the system during its entire life cycle, i.e., to even process it in encrypted form. This will significantly reduce the attack surface of the SlotMachine system and gives best prevention from data leakage, even from unintentional leakage of the platform operator due to misconfiguration or unknown software bugs. Additionally, protecting input data of competing airlines also forces honest bidding and prioritization of flights in the optimization process and prevents from collusion or bid rigging. Finally, to enable reliable and trustworthy data exchange between all participants of the platform we use blockchain technology. The blockchain enables as a trust anchor in the system and can be used to track all kinds of transactions in a robust manner. The blockchain serves as a (distributed) trust anchor for public data storage and can be used to track all kinds of transactions in a robust manner. It can also be used to manage the credit system used in SlotMachine to enable equity and fairness in the long run. All in all, the use of these technologies should increase the trustworthiness in to the SlotMachine platform thus leading to a higher participation of airlines ultimately leading to a more efficient operation of flights and airports. In the following we quickly introduce the core technologies mentioned, which were carefully studied in *D3.1* [9].

Multiparty computation (MPC) is a protocol between a defined number of players (or servers) holding secure inputs to securely compute some function *f* on these inputs. The security of the inputs should hold, even if players exhibit some amount of adversarial behaviour. The goal can be accomplished by an interactive protocol that the players execute. Intuitively, we want that the protocol execution is equivalent to having a trusted party *T* that receives the inputs privately, computes the function and returns the result, i.e., the MPC protocols emulates a trusted party in a distributed setting. With such a protocol we can - in principle - solve virtually any computation problem. The general theory of MPC was founded in the late 1980s and a huge body of research exists on the topic (see D3.1 – Report on State of the Art of Relevant Concepts [9]). However, until recently the concept was still considered impractical. Nevertheless, progress in the last decade culminated in many novel protocols which achieve good practical performance and can be used in real world applications.

Many different MPC protocols have been proposed in the past and they achieve very different properties and security guarantees. The main cryptographic properties achieved by a MPC protocol are correctness (if all parties behave honest the results of the computation is correct) and input privacy (a participating party does not learn anything about the inputs of other parties beyond what can be inferred from the result of the computation). In essence, MPC enables us to compute on encrypted data in a distributed setting, without decrypting them in the first place and the security of the input data for the computation is preserved as long as not more than a given thresholds of the servers are compromised by a single entity. In SlotMachine this technology will help to protect the privacy of sensitive data given by airspace users to the platform to conduct the global inter airline optimization process.

This component is the basic compute element used for privacy preserving data computation and is specified in detail in *D3.2* [3]. MPC nodes are controlled by the privacy engine and are instantiated and interconnected on a session basis to jointly conduct the real computation of the ranking. They only expose an interface via the privacy engine and are not intended to be used by any other component, they are also providing an encrypted channel to the AU client to receive sensitive information which must be hidden from all other components in the system. For this channel standard public key cryptography is used.

3.9 Blockchain

Relevant requirements: sm_4, co_7–8, rel_6

The design of the blockchain is still in development. A final decision requires additional experiments and discussions. In the following, we summarize the current state of our research.

A blockchain represents a distributed system that records transactions on a shared ledger. Its two main features are: consistency (ensures that all participants have the same view of the ledger) and immutability (ensures that if a transaction is accepted to the ledger, it cannot change any more). Distributed (shared) ledgers, especially those based on a blockchain, have been used for various purposes, initially as a provider of cryptocurrency but later also in different areas including academics, IoT, finance, industries, etc. From the emergence of Bitcoin in 2008, many development variants of this technology have emerged, e.g., Ethereum, various Hyperledger projects, Cardano, and others.

A blockchain-based system consists of network tasks on the lowest level, a consensus mechanism and a way of processing transactions including programming capabilities. The many blockchain frameworks differ in the strategy which was chosen for these three building blocks, especially the consensus mechanism and the transaction processing. With the consensus protocol, the nodes will be able to agree on a block which will be added to the immutable data structure. Blockchain-based systems rely on many cryptographic primitives and thereby provide a certain level of security. Often, they are also highly scalable, allowing for worldwide networks with hundreds of thousand nodes, but sometimes also smaller, closed networks are desirable.

The basic structure of a slot swapping service for a given airport naturally maps to a closed user group which maintains private information. The users of the system are rather static and known, therefore, a permissioned system seems to be most adequate. Furthermore, since we deal with a secure slot swapping service between competitive airlines, we assume that there can be adversaries among multiple parties participating in the network. Therefore, our system should be robust and cope with so called Byzantine failures, which addresses the challenges of potential adversaries in the network.

In SlotMachine, we aim to use the Tendermint framework for creating the SlotMachine blockchain. Every AU will also host their own blockchain client and may also run a blockchain node. The Tendermint blockchain is resistant to corruption on the provider side and also collusion of AUs as long as less than one third of the nodes are malicious.

3.10 Credit Wallets and Wallet Management

Relevant requirements: priv_11–14, fair_3, sm_2–4, co_7–8, au_4

The design of the credit wallet management is still in development. A final decision requires additional experiments and discussions. In the following, we summarize the current state of our research.

The credit wallets are maintained by a Wallet Management component that is attached to the Controller. The Controller, via the Wallet Management component, manages the wallets on behalf of the AUs. The credit balances are read-only for the AUs. AUs may view their own credit balances and changes of the credit balance in real-time but not those of others.

The AUs' credit balances are written regularly to the blockchain (see Section 3.9) but not instantly, i.e., not after every transaction but in accumulated form after certain intervals. The accumulated credit balances that are written on the blockchain are visible to every AU, but since that is not in real-time, the leakage of secrets is minimal. At the same time, allowing every AU to see how credit balances of the others change will likely increase the trust in the system's fairness through improved transparency. The system will be auditable if all parties collaborate and share raw data.

Note that Credit Wallets and Wallet Management are only required for realizing the market mechanisms SM1 and SM3 described in D2.3 – Business Concepts [3].

4 Interfaces

In this chapter, we describe REST and WebSocket interfaces of the functional components identified in Chapter 3.

4.1 Controller

POST /regulations/request

Request flight lists and regulations from NMF.

Parameter

This operation does not require parameters.

Example responses

201 Response

Responses

Status	Meaning	Description	Schema
201	Created	Created	string
400	Bad Request	Bad Request	string
404	Not Found	Not found	string

POST /regulations

Read flight lists and regulations from given data

Body parameter

```
<?xml version="1.0" encoding="UTF-8" ?>
<EnvelopeDTO>
  <body>
    <flightListByAerodromeReply>
      <requestReceptionTime>
        <text>2021-10-02 05:01:10</text>
      </requestReceptionTime>
      <requestId>R2C_ARD:16598542</requestId>
      <sendTime>
        <text>2021-10-02 05:01:11</text>
      </sendTime>
      <status>OK</status>
      <data>
        <effectiveTrafficWindow>
          <wef>2021-10-02 00:00</wef>
```

```
        <unt>2021-10-02 22:00</unt>
      </effectiveTrafficWindow>
      <flights>
        <flight>
          <flightId>FLIGHT1566</flightId>
          <keys>
            <aircraftId>AIRTXZS</aircraftId>
            <aerodromeOfDeparture>LSZH</aerodromeOfDeparture>
            <nonICAOAerodromeOfDeparture>
              false
            </nonICAOAerodromeOfDeparture>
            <airFiled>false</airFiled>
            <aerodromeOfDestination>DESM</aerodromeOfDestination>
            <nonICAOAerodromeOfDestination>
              false
            </nonICAOAerodromeOfDestination>
            <estimatedOffBlockTime>
              2021-10-02 13:05
            </estimatedOffBlockTime>
          </keys>
          <aircraftType>A319</aircraftType>
          <scheduledTakeOffTime>2021-10-02 13:20</scheduledTakeOffTime>
          <estimatedTakeOffTime>2021-10-02 13:20</estimatedTakeOffTime>
          <aircraftOperator>SWR</aircraftOperator>
          <operatingAircraftOperator>SWR</operatingAircraftOperator>
          <slotIssued>true</slotIssued>
          <delay>0</delay>
          <mostPenalisingRegulation>
            REGULATION1
          </mostPenalisingRegulation>
          <filedRegistrationMark>FRIOZ</filedRegistrationMark>
          <slotSwapCounter>
            <currentCounter>0</currentCounter>
            <maxLimit>3</maxLimit>
          </slotSwapCounter>
        </flight>
      </flights>
    </data>
  </flightListByAerodromeReply>
 </body>
</EnvelopeDTO>
```


Parameters

Name	In	Type	Required	Description
body	body	EnvelopeDTO	true	none

Example responses

201 Response


```
{
  "body": {
    "flightListByAerodromeReply": {
      "requestReceptionTime": {
        "text": "2021-10-02T05:01:00"
      },
      "requestId": "R2C_ARD:16598542",
      "sendTime": {
        "text": "2021-10-02T05:01:00"
      },
      "status": "OK",
      "data": {
        "effectiveTrafficWindow": {
          "wef": "2021-10-02 00:00",
          "unt": "2021-10-02 22:00"
        },
        "flights": [
          {
            "flight": {
              "flightId": {
                "flightId": "FLIGHT1566",
                "keys": {
                  "aircraftId": "AIRTXZS",
                  "aerodromeOfDeparture": "LSZH",
                  "nonICAOAerodromeOfDeparture": false,
                  "airFiled": false,
                  "aerodromeOfDestination": "DESM",
                  "nonICAOAerodromeOfDestination": false,
                  "estimatedOffBlockTime": {
                    "text": "2021-10-02T13:05:00"
                  }
                }
              },
              "aircraftType": "A319",
              "estimatedTakeOffTime": {
                "text": "2021-10-02T13:20:00"
              },
              "estimatedTimeOfArrival": {
                "text": "2021-10-02T14:29:00"
              },
              "aircraftOperator": "SWR",
              "operatingAircraftOperator": "SWR",
              "filedRegistrationMark": "FRIOZ",
              "slotSwapCounter": {
                "currentCounter": "0",
                "maxLimit": "3"
              },
              "calculatedTakeOffTime": {
```

```
            "text": "2021-10-02T13:20:00"
          },
          "mostPenalisingRegulation": "REGULATION1",
          "mostPenalisingRegulationCause": {
            "reason": "AERODROME_CAPACITY",
            "locationCategory": "DEPARTURE"
          }
        }
      }
    ]
  }
 }
}
```


The output is the representation of the given flight list data.

Responses

Status	Meaning	Description	Schema
201	Created	Created	EnvelopeDTO
400	Bad Request	Bad Request	string

`POST /optimizations/{optId}/start`

Start optimization process

Parameters

Name	In	Type	Required	Description
optId	path	string	true	none

Example responses

200 Response

Responses

Status	Meaning	Description	Schema
200	OK	OK	string
400	Bad Request	Bad Request	string
404	Not Found	Not found	string

POST /optimizations/{optId}/solutions/propose

Propose flight list to NMF

Parameters

Name	In	Type	Required	Description
optId	path	string	true	none

Example responses

200 Response


```
{
  "regulationId": "REGULATION1",
  "optimizationId": "119c4ba1-0895-4fec-bf18-23273d64b8d7",
  "currentServerTime": "2019-08-24 14:15",
  "rejectUntil": "2019-08-24 14:15",
  "solutions": [
    {
      "regulationId": "REGULATION1",
      "optimizationId": "119c4ba1-0895-4fec-bf18-23273d64b8d7",
      "solutionId": "SOLUTION1",
      "priority": 1,
      "flights": [
        {
          "flightId": "FLIGHT1566",
          "slotTime": "2019-08-24 14:15"
        }
      ]
    }
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	OK	OK	SolutionListDTO
400	Bad Request	Bad Request	string
404	Not Found	Not found	SolutionListDTO

POST /registrations

Register airspace user for regulations at an airport

Register an airspace user's client for regulations at an airport; overwrite existing registration of an airspace user if already exists.

Body parameter

```
{
  "airlineId": "SWR",
  "airportId": "LSZH",
  "regulationType": "DEPARTURE",
  "callbackEndpoint": "http://localhost:8080/callbackEndpoint"
}
```

Parameters

Name	In	Type	Required	Description
body	body	RegulationRegistrationDTO	true	none

Example responses

200 Response

```
{
  "airlineId": "SWR",
  "airportId": "LSZH",
  "regulationType": "DEPARTURE",
  "callbackEndpoint": "http://localhost:8080/callbackEndpoint"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Resource updated	RegulationRegistrationDTO
201	Created	Resource created	RegulationRegistrationDTO
400	Bad Request	Invalid input supplied	None
default	Default	The registration	RegulationRegistrationDTO

`POST /optimizations/{optId}/solutions/{solutionId}/rejection/{airlineId}`

Reject a solution found by an optimization run

Parameters

Name	In	Type	Required	Description
optId	path	string	true	none
solutionId	path	string	true	none
airlineId	path	string	true	none

Example responses

200 Response


```
{
  "airlineId": "SWR",
  "regulationId": "LSZH",
  "optimizationId": " 119c4ba1-0895-4fec-bf18-23273d64b8d7",
  "solutionId": "SOLUTION1"
}
```


Responses

Status	Meaning	Description	Schema
200	OK	OK	RejectedSolutionDTO
400	Bad Request	Bad Request	string
404	Not Found	Not found	RejectedSolutionDTO

POST /optimizations/{optId}/solutions/rejections

Reject multiple solutions found by an optimization run

Body parameter


```
{
  "airlineId": "SWR",
  "regulationId": "REGULATION1",
  "optimizationId": " 119c4ba1-0895-4fec-bf18-23273d64b8d7",
  "rejectedSolutions": [
    {
      "airlineId": "SWR",
      "regulationId": "REGULATION1",
      "optimizationId": " 119c4ba1-0895-4fec-bf18-23273d64b8d7",
      "solutionId": "SOLUTION1"
    }
  ]
}
```


Parameters

Name	In	Type	Required	Description
body	body	RejectedSolutionListDTO	true	none

Example responses

200 Response


```
{
  "airlineId": "SWR",
  "regulationId": "REGULATION1",
  "optimizationId": " 119c4ba1-0895-4fec-bf18-23273d64b8d7",
```


```
  "rejectedSolutions": [
    {
      "airlineId": "SWR",
      "regulationId": "REGULATION1",
      "optimizationId": " 119c4ba1-0895-4fec-bf18-23273d64b8d7",
      "solutionId": "SOLUTION1"
    }
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	OK	OK	RejectedSolutionListDTO
400	Bad Request	Bad Request	string
404	Not Found	Not found	RejectedSolutionListDTO

POST /optimizations/{optId}/solutions/accept

Notify the SlotMachine of the accepted flight list

Body parameter


```
{
  "optimizationId": "119c4ba1-0895-4fec-bf18-23273d64b8d7",
  "solutionId": "SOLUTION1"
}
```


Parameters

Name	In	Type	Required	Description
optId	path	string	true	none
body	body	AcceptedFlightListDTO	true	none

Example responses

200 Response


```
{
  "optimizationId": "119c4ba1-0895-4fec-bf18-23273d64b8d7",
  "solutionId": "SOLUTION1"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	OK	AcceptedFlightListDTO
400	Bad Request	Bad Request	string
404	Not Found	Not found	AcceptedFlightListDTO

`POST /optimizations/{optId}/preferences/{airlineId}`

Submit slot preferences for flights

Submit slot preferences for flights of an airspace user.

Body parameter

```
[
  {
    "regulationId": "REGULATION1",
    "optimizationId": "119c4ba1-0895-4fec-bf18-23273d64b8d7",
    "airlineId": "SWR",
    "flights": [
      {
        "flightId": "FLIGHT1566",
        "weightMap": [
          {
            "slotTime": "2021-10-02 13:20",
            "weight": "100"
          }
        ]
      }
    ]
  }
]
```

Parameters

Name	In	Type	Required	Description
optimizationId	path	string	true	none
airlineId	path	string	true	none
body	body	WeightMapDTO	true	none

Example responses

200 Response


```
[
  {
    "regulationId": "REGULATION1",
    "optimizationId": "119c4ba1-0895-4fec-bf18-23273d64b8d7",
    "airlineId": "SWR",
    "flights": [
      {
        "flightId": "FLIGHT1566",
        "weightMap": [
          {
            "slotTime": "2021-10-02 13:20",
            "weight": "100"
          }
        ]
      }
    ]
  }
]
```


Responses

Status	Meaning	Description	Schema
200	OK	OK	Inline
400	Bad Request	Bad Request	string
404	Not Found	Not found	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	[WeightMapDTO]	false	none	
» regulationId	string	false	none	none
» optimizationId	string	false	none	none
» airlineId	string	false	none	none
» flights	[WeightMapFlightDTO]	false	none	none
»» flightId	string	false	none	none
»» weightMap	[WeightMapObjectDTO]	false	none	none
»»» slotTime	string(date-time)	false	none	none
»»» weight	string	false	none	none

Status Code **404**

Name	Type	Required	Restrictions	Description
anonymous	[WeightMapDTO]	false	none	none
» regulationId	string	false	none	none
» optimizationId	string	false	none	none
» airlineId	string	false	none	none
» flights	[WeightMapFlightDTO]	false	none	none
»» flightId	string	false	none	none
»» weightMap	[WeightMapObjectDTO]	false	none	none
»»» slotTime	string(date-time)	false	none	none
»»» weight	string	false	none	none

`POST /optimizations/{optId}/preferences`

Submit slot preferences for flights

Submit slot preferences for flights of an airspace user.

Body parameter

```
[
  {
    "regulationId": "string",
    "optimizationId": "string",
    "airlineId": "string",
    "flights": [
      {
        "flightId": "string",
        "weightMap": [
          {
            "slotTime": "2019-08-24T14:15:22Z",
            "weight": "string"
          }
        ]
      }
    ]
  }
]
```

Parameters

Name	In	Type	Required	Description
optimizationId	path	string	true	none
body	body	WeightMapDTO	true	none

Example responses

200 Response


```
[
  {
    "regulationId": "string",
    "optimizationId": "string",
    "airlineId": "string",
    "flights": [
      {
        "flightId": "string",
        "weightMap": [
          {
            "slotTime": "2019-08-24T14:15:22Z",
            "weight": "string"
          }
        ]
      }
    ]
  }
]
```


Responses

Status	Meaning	Description	Schema
200	OK	OK	Inline
400	Bad Request	Bad Request	string
404	Not Found	Not found	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	[WeightMapDTO]	false	none	none
» regulationId	string	false	none	none
» optimizationId	string	false	none	none
» airlineId	string	false	none	none
» flights	[WeightMapFlightDTO]	false	none	none
»» flightId	string	false	none	none
»» weightMap	[WeightMapObjectDTO]	false	none	none
»»» slotTime	string(date-time)	false	none	none
»»» weight	string	false	none	none

Status Code **404**

Name	Type	Required	Restrictions	Description
anonymous	[WeightMapDTO]	false	none	none
» regulationId	string	false	none	none
» optimizationId	string	false	none	none
» airlineId	string	false	none	none
» flights	[WeightMapFlightDTO]	false	none	none
»» flightId	string	false	none	none
»» weightMap	[WeightMapObjectDTO]	false	none	none
»»» slotTime	string(date-time)	false	none	none
»»» weight	string	false	none	none

POST /optimizations/preferences

Submit slot preferences for flights

Submit slot preferences for flights of an airspace user.

Body parameter

```
[
  {
    "regulationId": "string",
    "optimizationId": "string",
    "airlineId": "string",
    "flights": [
      {
        "flightId": "string",
        "weightMap": [
          {
            "slotTime": "2019-08-24T14:15:22Z",
            "weight": "string"
          }
        ]
      }
    ]
  }
]
```

Parameters

Name	In	Type	Required	Description
body	body	WeightMapDTO	true	none

Example responses

200 Response


```
[
  {
    "regulationId": "string",
    "optimizationId": "string",
    "airlineId": "string",
    "flights": [
      {
        "flightId": "string",
        "weightMap": [
          {
            "slotTime": "2019-08-24T14:15:22Z",
            "weight": "string"
          }
        ]
      }
    ]
  }
]
```


Responses

Status	Meaning	Description	Schema
200	OK	OK	Inline
400	Bad Request	Bad Request	string
404	Not Found	Not found	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	[WeightMapDTO]	false	none	none
» regulationId	string	false	none	none
» optimizationId	string	false	none	none
» airlineId	string	false	none	none
» flights	[WeightMapFlightDTO]	false	none	none
»» flightId	string	false	none	none
»» weightMap	[WeightMapObjectDTO]	false	none	none
»»» slotTime	string(date-time)	false	none	none
»»» weight	string	false	none	none

Status Code **404**

Name	Type	Required	Restrictions	Description
anonymous	[WeightMapDTO]	false	none	none
» regulationId	string	false	none	none
» optimizationId	string	false	none	none
» airlineId	string	false	none	none
» flights	[WeightMapFlightDTO]	false	none	none
»» flightId	string	false	none	none
»» weightMap	[WeightMapObjectDTO]	false	none	none
»»» slotTime	string(date-time)	false	none	none
»»» weight	string	false	none	none

`GET /registrations/{airlineId}/{airportId}/optimizations/current/flightList`

Get current optimization

Get information about the current optimization for an airspace user's registration.

Parameters

Name	In	Type	Required	Description
airlineId	path	string	true	none
airportId	path	string	true	none

Example responses

200 Response


```
{
  "optimization": {
    "regulationId": "string",
    "requestReceptionTime": "2019-08-24T14:15:22Z",
    "requestId": "string",
    "optimizationId": "string",
    "state": "WAIT_FOR_INPUTS",
    "currentServerTime": "2019-08-24T14:15:22Z",
    "cutOffTime": "2019-08-24T14:15:22Z",
    "nextOptimizationRun": "2019-08-24T14:15:22Z"
  },
  "slots": [
    {
      "slotTime": "2019-08-24T14:15:22Z"
    }
  ],
```

```
  "flights": [
    {
      "flightId": "string",
      "keys": {
        "aircraftId": "string",
        "aerodromeOfDeparture": "string",
        "nonICAOAerodromeOfDeparture": true,
        "airFiled": true,
        "aerodromeOfDestination": "string",
        "nonICAOAerodromeOfDestination": true,
        "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
      },
      "aircraftType": "string",
      "scheduledTakeOffTime": "2019-08-24T14:15:22Z",
      "estimatedTakeOffTime": "2019-08-24T14:15:22Z",
      "aircraftOperator": "string",
      "operatingAircraftOperator": "string",
      "slotIssued": true,
      "delay": 0,
      "mostPenalisingRegulation": "string",
      "filedRegistrationMark": "string",
      "slotSwapCounter": {
        "currentCounter": 0,
        "maxLimit": 0
      }
    }
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	OK	OK	FlightListDTO
400	Bad Request	Bad Request	string
404	Not Found	Not found	FlightListDTO

GET /optimizations/{optId}/solutions/accepted

Get accepted solution of optimization runs

Parameters

Name	In	Type	Required	Description
optId	path	string	true	none

Example responses

200 Response


```
{
  "regulationId": "string",
  "optimizationId": "string",
  "currentServerTime": "2019-08-24T14:15:22Z",
  "rejectUntil": "2019-08-24T14:15:22Z",
  "solutions": [
    {
      "regulationId": "string",
      "optimizationId": "string",
      "solutionId": "string",
      "priority": 0,
      "flights": [
        {
          "flightId": "string",
          "slotTime": "2019-08-24T14:15:22Z"
        }
      ]
    }
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	OK	accepted solution	SolutionListDTO
400	Bad Request	Bad Request	string
404	Not Found	Not found	SolutionListDTO

GET /optimizations/{optId}/solutions

Get results of optimization runs

Parameters

Name	In	Type	Required	Description
optId	path	string	true	none

Example responses

200 Response


```
{
  "regulationId": "string",
```

```
  "optimizationId": "string",
  "currentServerTime": "2019-08-24T14:15:22Z",
  "rejectUntil": "2019-08-24T14:15:22Z",
  "solutions": [
    {
      "regulationId": "string",
      "optimizationId": "string",
      "solutionId": "string",
      "priority": 0,
      "flights": [
        {
          "flightId": "string",
          "slotTime": "2019-08-24T14:15:22Z"
        }
      ]
    }
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	OK	Array of optimization results	SolutionListDTO
400	Bad Request	Bad Request	string
404	Not Found	Not found	SolutionListDTO

`DELETE /registrations/{airlineId}/{airportId}`

Delete an airspace user client's registration for regulations at an airport

Parameters

Name	In	Type	Required	Description
airlineId	path	string	true	none
airportId	path	string	true	none

Example responses

200 Response


```
{
  "airlineId": "string",
  "airportId": "string",
  "regulationType": "DEPARTURE",
  "callbackEndpoint": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Has been deleted	RegulationRegistrationDTO
400	Bad Request	Bad Request	string
404	Not Found	Not found	RegulationRegistrationDTO

`GET /controller/active`

Check if controller is active

Responses

Status	Meaning	Description	Schema
200	OK	OK	string
400	Bad Request	Bad Request	string

Schemas

In the following, we describe the schema of the data transfer objects.

BodyDTO

```
{
  "flightListByAerodromeReply": {
    "requestReceptionTime": {
      "text": "2019-08-24T14:15:22Z"
    },
    "requestId": "string",
    "sendTime": {
      "text": "2019-08-24T14:15:22Z"
    },
    "status": "string",
    "data": {
      "effectiveTrafficWindow": {
        "wef": "string",
        "unt": "string"
      },
      "flights": [
        {
          "flight": {
            "flightId": "string",
            "keys": {
              "aircraftId": "string",
```

```
            "aerodromeOfDeparture": "string",
            "nonICAOAerodromeOfDeparture": true,
            "airFiled": true,
            "aerodromeOfDestination": "string",
            "nonICAOAerodromeOfDestination": true,
            "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
          },
          "aircraftType": "string",
          "scheduledTakeOffTime": "2019-08-24T14:15:22Z",
          "estimatedTakeOffTime": "2019-08-24T14:15:22Z",
          "aircraftOperator": "string",
          "operatingAircraftOperator": "string",
          "slotIssued": true,
          "delay": 0,
          "mostPenalisingRegulation": "string",
          "filedRegistrationMark": "string",
          "slotSwapCounter": {
            "currentCounter": 0,
            "maxLimit": 0
          }
        }
      }
    ]
  }
 }
}
```


Name	Type	Required	Description
flightListByAerodromeReply	FlightListByAerodromeReplyDTO	false	none

DataDTO


```
{
  "effectiveTrafficWindow": {
    "wef": "string",
    "unt": "string"
  },
  "flights": [
    {
      "flight": {
        "flightId": "string",
        "keys": {
          "aircraftId": "string",
          "aerodromeOfDeparture": "string",
          "nonICAOAerodromeOfDeparture": true,
          "airFiled": true,
          "aerodromeOfDestination": "string",
          "nonICAOAerodromeOfDestination": true,
          "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
```

```
        },
        "aircraftType": "string",
        "scheduledTakeOffTime": "2019-08-24T14:15:22Z",
        "estimatedTakeOffTime": "2019-08-24T14:15:22Z",
        "aircraftOperator": "string",
        "operatingAircraftOperator": "string",
        "slotIssued": true,
        "delay": 0,
        "mostPenalisingRegulation": "string",
        "filedRegistrationMark": "string",
        "slotSwapCounter": {
          "currentCounter": 0,
          "maxLimit": 0
        }
      }
    }
  ]
}
```


Name	Type	Required	Description
effectiveTrafficWindow	EffectiveTrafficWindowDTO	false	none
flights	[FlightsDTO]	false	none

EffectiveTrafficWindowDTO


```
{
  "wef": "string",
  "unt": "string"
}
```


Name	Type	Required	Restrictions	Name
wef	string	false	none	wef
unt	string	false	none	none

EnvelopeDTO


```
{
  "body": {
    "flightListByAerodromeReply": {
      "requestReceptionTime": {
        "text": "2019-08-24T14:15:22Z"
      },
      "requestId": "string",
      "sendTime": {
        "text": "2019-08-24T14:15:22Z"
      },
      "status": "string",
      "data": {
```

```
        "effectiveTrafficWindow": {
          "wef": "string",
          "unt": "string"
        },
        "flights": [
          {
            "flight": {
              "flightId": "string",
              "keys": {
                "aircraftId": "string",
                "aerodromeOfDeparture": "string",
                "nonICAOAerodromeOfDeparture": true,
                "airFiled": true,
                "aerodromeOfDestination": "string",
                "nonICAOAerodromeOfDestination": true,
                "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
              },
              "aircraftType": "string",
              "scheduledTakeOffTime": "2019-08-24T14:15:22Z",
              "estimatedTakeOffTime": "2019-08-24T14:15:22Z",
              "aircraftOperator": "string",
              "operatingAircraftOperator": "string",
              "slotIssued": true,
              "delay": 0,
              "mostPenalisingRegulation": "string",
              "filedRegistrationMark": "string",
              "slotSwapCounter": {
                "currentCounter": 0,
                "maxLimit": 0
              }
            }
          }
        ]
      }
    }
  }
}
```


Name	Type	Required	Description
body	BodyDTO	false	

FlightDTO


```
{
  "flightId": "string",
  "keys": {
    "aircraftId": "string",
    "aerodromeOfDeparture": "string",
    "nonICAOAerodromeOfDeparture": true,
```

```
    "airFiled": true,
    "aerodromeOfDestination": "string",
    "nonICAOAerodromeOfDestination": true,
    "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
  },
  "aircraftType": "string",
  "scheduledTakeOffTime": "2019-08-24T14:15:22Z",
  "estimatedTakeOffTime": "2019-08-24T14:15:22Z",
  "aircraftOperator": "string",
  "operatingAircraftOperator": "string",
  "slotIssued": true,
  "delay": 0,
  "mostPenalisingRegulation": "string",
  "filedRegistrationMark": "string",
  "slotSwapCounter": {
    "currentCounter": 0,
    "maxLimit": 0
  }
}
```


Name	Type	Required	Description
flightId	string	false	
keys	FlightKeyDTO	false	
aircraftType	string	false	
scheduledTakeOffTime	string(date-time)	false	
estimatedTakeOffTime	string(date-time)	false	
aircraftOperator	string	false	
operatingAircraftOperator	string	false	
slotIssued	boolean	false	
delay	integer(int32)	false	
mostPenalisingRegulation	string	false	
filedRegistrationMark	string	false	
slotSwapCounter	SlotSwapCounterDTO	false	

FlightListByAerodromeReplyDTO


```
{
  "requestReceptionTime": {
    "text": "2019-08-24T14:15:22Z"
  },
  "requestId": "string",
  "sendTime": {
    "text": "2019-08-24T14:15:22Z"
  },
  "status": "string",
  "data": {
```

```
    "effectiveTrafficWindow": {
      "wef": "string",
      "unt": "string"
    },
    "flights": [
      {
        "flight": {
          "flightId": "string",
          "keys": {
            "aircraftId": "string",
            "aerodromeOfDeparture": "string",
            "nonICAOAerodromeOfDeparture": true,
            "airFiled": true,
            "aerodromeOfDestination": "string",
            "nonICAOAerodromeOfDestination": true,
            "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
          },
          "aircraftType": "string",
          "scheduledTakeOffTime": "2019-08-24T14:15:22Z",
          "estimatedTakeOffTime": "2019-08-24T14:15:22Z",
          "aircraftOperator": "string",
          "operatingAircraftOperator": "string",
          "slotIssued": true,
          "delay": 0,
          "mostPenalisingRegulation": "string",
          "filedRegistrationMark": "string",
          "slotSwapCounter": {
            "currentCounter": 0,
            "maxLimit": 0
          }
        }
      }
    ]
  }
}
```


Name	Type	Required	Description
requestReceptionTime	LocalDateTimeToMinOrSecDTO	false	
requestId	string	false	
sendTime	LocalDateTimeToMinOrSecDTO	false	
status	string	false	
data	DataDTO	false	

FlightsDTO


```
{
  "flight": {
    "flightId": "string",
```

```
    "keys": {
      "aircraftId": "string",
      "aerodromeOfDeparture": "string",
      "nonICAOAerodromeOfDeparture": true,
      "airFiled": true,
      "aerodromeOfDestination": "string",
      "nonICAOAerodromeOfDestination": true,
      "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
    },
    "aircraftType": "string",
    "scheduledTakeOffTime": "2019-08-24T14:15:22Z",
    "estimatedTakeOffTime": "2019-08-24T14:15:22Z",
    "aircraftOperator": "string",
    "operatingAircraftOperator": "string",
    "slotIssued": true,
    "delay": 0,
    "mostPenalisingRegulation": "string",
    "filedRegistrationMark": "string",
    "slotSwapCounter": {
      "currentCounter": 0,
      "maxLimit": 0
    }
  }
}
```


Name	Type	Required	Description
flight	FlightDTO	false	

LocalDateTimeToMinOrSecDTO


```
{
  "text": "2019-08-24T14:15:22Z"
}
```


Name	Type	Required	Description
text	string(date-time)	false	

SlotSwapCounterDTO


```
{
  "currentCounter": 0,
  "maxLimit": 0
}
```


Name	Type	Required	Description
currentCounter	integer(int32)	false	
maxLimit	integer(int32)	false	

RegulationRegistrationDTO

```
{
  "airlineId": "string",
  "airportId": "string",
  "regulationType": "DEPARTURE",
  "callbackEndpoint": "string"
}
```

Name	Type	Required	Description
airlineId	string	false	
airportId	string	false	
regulationType	string	false	
callbackEndpoint	string	false	

Property	Value
regulationType	DEPARTURE
regulationType	ARRIVAL
regulationType	EN_ROUTE

RejectedSolutionDTO

```
{
  "airlineId": "string",
  "regulationId": "string",
  "optimizationId": "string",
  "solutionId": "string"
}
```

Name	Type	Required	Description
airlineId	string	false	
regulationId	string	false	
optimizationId	string	false	
solutionId	string	false	

RejectedSolutionListDTO

```
{
  "airlineId": "string",
  "regulationId": "string",
  "optimizationId": "string",
  "rejectedSolutions": [
    {
```

```
      "airlineId": "string",
      "regulationId": "string",
      "optimizationId": "string",
      "solutionId": "string"
    }
  ]
}
```


Name	Type	Required	Description
airlineId	string	false	
regulationId	string	false	
optimizationId	string	false	
rejectedSolutions	[RejectedSolutionDTO]	false	

SolutionDTO


```
{
  "regulationId": "string",
  "optimizationId": "string",
  "solutionId": "string",
  "priority": 0,
  "flights": [
    {
      "flightId": "string",
      "slotTime": "2019-08-24T14:15:22Z"
    }
  ]
}
```


Name	Type	Required	Description
regulationId	string	false	
optimizationId	string	false	
solutionId	string	false	
priority	integer(int32)	false	
flights	[SolutionFlightDTO]	false	

SolutionFlightDTO


```
{
  "flightId": "string",
  "slotTime": "2019-08-24T14:15:22Z"
}
```

Name	Type	Required	Description
flightId	string	false	
slotTime	string(date-time)	false	

SolutionListDTO

```
{
  "regulationId": "string",
  "optimizationId": "string",
  "currentServerTime": "2019-08-24T14:15:22Z",
  "rejectUntil": "2019-08-24T14:15:22Z",
  "solutions": [
    {
      "regulationId": "string",
      "optimizationId": "string",
      "solutionId": "string",
      "priority": 0,
      "flights": [
        {
          "flightId": "string",
          "slotTime": "2019-08-24T14:15:22Z"
        }
      ]
    }
  ]
}
```

Name	Type	Required	Description
regulationId	string	false	
optimizationId	string	false	
currentServerTime	string(date-time)	false	
rejectUntil	string(date-time)	false	
solutions	[SolutionDTO]	false	

AcceptedFlightListDTO

```
{
  "optimizationId": "string",
  "solutionId": "string"
}
```

Name	Type	Required	Description
optimizationId	string	false	
solutionId	string	false	

WeightMapDTO

```
{
  "regulationId": "string",
  "optimizationId": "string",
  "airlineId": "string",
  "flights": [
    {
      "flightId": "string",
      "weightMap": [
        {
          "slotTime": "2019-08-24T14:15:22Z",
          "weight": "string"
        }
      ]
    }
  ]
}
```

Name	Type	Required	Description
regulationId	string	false	
optimizationId	string	false	
airlineId	string	false	
flights	[WeightMapFlightDTO]	false	

WeightMapFlightDTO

```
{
  "flightId": "string",
  "weightMap": [
    {
      "slotTime": "2019-08-24T14:15:22Z",
      "weight": "string"
    }
  ]
}
```

Name	Type	Required	Description
flightId	string	false	
weightMap	[WeightMapObjectDTO]	false	

WeightMapObjectDTO

```
{
  "slotTime": "2019-08-24T14:15:22Z",
  "weight": "string"
}
```

Name	Type	Required	Description
slotTime	string(date-time)	false	
weight	string	false	

FlightKeyDTO


```
{
  "aircraftId": "string",
  "aerodromeOfDeparture": "string",
  "nonICAOAerodromeOfDeparture": true,
  "airFiled": true,
  "aerodromeOfDestination": "string",
  "nonICAOAerodromeOfDestination": true,
  "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
}
```


Name	Type	Required	Description
aircraftId	string	false	
aerodromeOfDeparture	string	false	
nonICAOAerodromeOfDeparture	boolean	false	
airFiled	boolean	false	
aerodromeOfDestination	string	false	
nonICAOAerodromeOfDestination	boolean	false	
estimatedOffBlockTime	string(date-time)	false	

FlightListDTO


```
{
  "optimization": {
    "regulationId": "string",
    "requestReceptionTime": "2019-08-24T14:15:22Z",
    "requestId": "string",
    "optimizationId": "string",
    "state": "WAIT_FOR_INPUTS",
    "currentServerTime": "2019-08-24T14:15:22Z",
    "cutOffTime": "2019-08-24T14:15:22Z",
    "nextOptimizationRun": "2019-08-24T14:15:22Z"
  },
  "slots": [
    {
      "slotTime": "2019-08-24T14:15:22Z"
    }
  ],
  "flights": [
    {
      "flightId": "string",
```

```
      "keys": {
        "aircraftId": "string",
        "aerodromeOfDeparture": "string",
        "nonICAOAerodromeOfDeparture": true,
        "airFiled": true,
        "aerodromeOfDestination": "string",
        "nonICAOAerodromeOfDestination": true,
        "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
      },
      "aircraftType": "string",
      "scheduledTakeOffTime": "2019-08-24T14:15:22Z",
      "estimatedTakeOffTime": "2019-08-24T14:15:22Z",
      "aircraftOperator": "string",
      "operatingAircraftOperator": "string",
      "slotIssued": true,
      "delay": 0,
      "mostPenalisingRegulation": "string",
      "filedRegistrationMark": "string",
      "slotSwapCounter": {
        "currentCounter": 0,
        "maxLimit": 0
      }
    }
  ]
}
```


Name	Type	Required	Description
optimization	OptimizationDTO	false	
slots	[SlotDTO]	false	
flights	[FlightDTO]	false	

OptimizationDTO


```
{
  "regulationId": "string",
  "requestReceptionTime": "2019-08-24T14:15:22Z",
  "requestId": "string",
  "optimizationId": "string",
  "state": "WAIT_FOR_INPUTS",
  "currentServerTime": "2019-08-24T14:15:22Z",
  "cutOffTime": "2019-08-24T14:15:22Z",
  "nextOptimizationRun": "2019-08-24T14:15:22Z"
}
```

Name	Type	Required	Description
regulationId	string	false	
requestReceptionTime	string(date-time)	false	
requestId	string	false	
optimizationId	string	false	
state	string	false	
currentServerTime	string(date-time)	false	
cutOffTime	string(date-time)	false	
nextOptimizationRun	string(date-time)	false	

Property	Value
state	WAIT_FOR_INPUTS
state	CUT_OFF_TIME_REACHED
state	OPTIMIZATION_RUNNING
state	WAITING_FOR_REJECTS
state	WAITING_FOR_NM_CONFIRMATION
state	FINISHED

SlotDTO

```
{
  "slotTime": "2019-08-24T14:15:22Z"
}
```

Name	Type	Required	Restrictions	Description
slotTime	string(date-time)	false	none	none

4.2 Heuristic Optimizer

`POST /optimizations`

Create and initialize a (heuristic) optimization for flights and slots.

Body parameter

```
{
  "flights": [
    {
      "flightId": "F1",
      "scheduledTime": "2019-08-24T14:15:22Z",
    }
  ],
```

```
  "initialFlightSequence": [
    "F1", "F2", "F3"
  ],
  "optId": "0d6b6665-5948-4218-bf74-bb86e549040e",
  "optimizationFramework": "JENETICS",
  "optimizationMode": "PRIVACY_PRESERVING",
  "fitnessEstimator": "LINEAR",
  "parameters": {},
  "privacyEngineEndpoint": "https://example.com/privacy_engine",
  "slots": [
    {
      "time": "2019-08-24T14:15:22Z"
    }
  ]
}
```


We refer to D4.2 – Specification of Evolutionary Algorithm [4] for more detailed specifications of the parameters. Please note the following regarding the parameters.

- The “parameters” map consists of key/value pairs that may contain additional parameters for the optimization run that are specific to the used optimization framework, e.g., Jenetics. If omitted, default values will be used.
- The Heuristic Optimizer also allows to pass margins and weight maps for running the optimizer in a non-privacy-preserving mode that does not involve a Privacy Engine. This should not be confused with the non-privacy-preserving demonstrator iteration of the SlotMachine system but rather serves for development and demonstration purposes but could also serve as the basis for a SlotMachine system that supports multi-objective optimization without a Privacy Engine.

Parameters

Name	In	Type	Required	Description
body	body	OptimizationDTO	true	optimization

Example responses

201 Response


```
{
  "flights": [
    {
      "flightId": "F1",
      "scheduledTime": "2019-08-24T14:15:22Z",
    }
  ],
  "initialFlightSequence": [
    "F1", "F2", "F3"
  ],
```

```
  "optId": "0d6b6665-5948-4218-bf74-bb86e549040e",
  "optimizationFramework": "JENETICS",
  "optimizationMode": "PRIVACY_PRESERVING",
  "fitnessEstimator": "LINEAR",
  "optimizationStatus": "CREATED",
  "parameters": {},
  "privacyEngineEndpoint": "https://10.0.0.138/privacy_engine/",
  "slots": [
    {
      "time": "2019-08-24T14:15:22Z"
    }
  ],
  "timestamp": "2019-08-24T14:15:22Z"
}
```


The output is a representation of the optimization run that was just created. The optimization status is set to “CREATED”. The timestamp shows the date and time that the request was made, indicating at which point in time the data about the optimization run was current.

Responses

Status	Meaning	Description	Schema
200	OK	OK	OptimizationDTO
201	Created	Created	OptimizationDTO
400	Bad Request	Bad Request	None
401	Unauthorized	Unauthorized	None
403	Forbidden	Forbidden	None
404	Not Found	Not Found	None

`GET /optimizations/{optId}`

Get the description of a specific optimization.

Parameters

Name	In	Type	Required	Description
optId	path	string	true	the optimization’s identifier

Example responses

200 Response


```
{
  "flights": [
```

```
    {
      "flightId": "F1",
      "scheduledTime": "2019-08-24T14:15:22Z",
    }
  ],
  "initialFlightSequence": [
    "F1"
  ],
  "optId": "0d6b6665-5948-4218-bf74-bb86e549040e",
  "optimizationFramework": "JENETICS",
  "optimizationMode": "PRIVACY_PRESERVING",
  "fitnessEstimator": "LINEAR",
  "optimizationStatus": "INITIALIZED",
  "parameters": {},
  "privacyEngineEndpoint": "https://10.0.0.138/privacy_engine/",
  "slots": [
    {
      "time": "2019-08-24T14:15:22Z"
    }
  ],
  "timestamp": "2019-08-24T14:15:22Z"
}
```


Responses

Status	Meaning	Description	Schema
200	OK	OK	OptimizationDTO
401	Unauthorized	Unauthorized	None
403	Forbidden	Forbidden	None
404	Not Found	Not Found	None

`DELETE /optimizations/{optId}`

Delete an optimization and its results, if available. Abort a running optimization.

Parameters

Name	In	Type	Required	Description
optId	path	string	true	the optimization's identifier

Responses

Status	Meaning	Description	Schema
200	OK	OK; the deletion worked	None
204	No Content	No Content	None
401	Unauthorized	Unauthorized	None
403	Forbidden	Forbidden	None

Status	Meaning	Description	Schema
404	Not Found	Not Found; no optimization with the specified identifier exists	None

PUT /optimizations/{optId}/start

Start a specific optimization that was previously created and initialized.

Parameters

Name	In	Type	Required	Description
optId	path	string	true	the optimization's identifier

Example responses

202 Response


```
{
  "fitnessEstimator": "LOGARITHMIC",
  "flights": [
    {
      "flightId": "F01",
      "margins": {
        "scheduledTime": "2019-08-24T14:15:22Z",
        "timeNotAfter": "2019-08-24T14:15:22Z",
        "timeNotBefore": "2019-08-24T14:15:22Z",
        "timeWished": "2019-08-24T14:15:22Z"
      },
      "scheduledTime": "2019-08-24T14:15:22Z",
      "weightMap": [
        10, 10, 100
      ]
    }
  ],
  "initialFlightSequence": [
    "string"
  ],
  "optId": "string",
  "optimizationFramework": "string",
  "optimizationMode": "PRIVACY_PRESERVING",
  "optimizationStatus": "RUNNING",
  "parameters": {},
  "privacyEngineEndpoint": "string",
  "slots": [
    {
      "time": "2019-08-24T14:15:22Z"
    }
```

```
  ],
  "timestamp": "2019-08-24T14:15:22Z"
}
```


Responses

Status	Meaning	Description	Schema
200	OK	OK; the optimization is already running.	OptimizationDTO
202	Accepted	Accepted; if the optimization was successfully started.	OptimizationDTO
303	See Other	See Other; returns location of result in header, if cancelled or already done.	None
401	Unauthorized	Unauthorized	None
403	Forbidden	Forbidden	None
404	Not Found	Not Found	None

`PUT /optimizations/{optId}/abort`

Abort a previously started optimization; if available, an intermediate result can be obtained.

Parameters

Name	In	Type	Required	Description
optId	path	string	true	the optimization's identifier

Example responses

200 Response


```
{
  "fitnessEstimator": "string",
  "flights": [
    {
      "flightId": "F1",
      "scheduledTime": "2019-08-24T14:15:22Z",
    }
  ],
  "initialFlightSequence": [
    "F1"
  ],
  "optId": "0d6b6665-5948-4218-bf74-bb86e549040e",
  "optimizationFramework": "JENETICS",
  "optimizationMode": "PRIVACY_PRESERVING",
  "optimizationStatus": "CANCELLED",
  "parameters": {},
  "privacyEngineEndpoint": "https://10.0.0.138/privacy_engine/",
  "slots": [
```

```
    {
      "time": "2019-08-24T14:15:22Z"
    }
  ],
  "timestamp": "2019-08-24T14:15:22Z"
}
```


Responses

Status	Meaning	Description	Schema
200	OK	OK; optimization run successfully aborted.	OptimizationDTO
201	Created	Created	None
401	Unauthorized	Unauthorized	None
403	Forbidden	Forbidden	None
404	Not Found	Not Found; no optimization with the specified identifier exists.	None

PUT /optimizations/{optId}/start/wait

Run a previously created and initialized optimization in a synchronized way, waiting for the response.

Parameters

Name	In	Type	Required	Description
optId	path	string	true	the optimization's identifier

Example responses

200 Response


```
{
  "fitness": 0,
  "fitnessFunctionInvocations": 0,
  "optId": "string",
  "optimizedFlightSequence": [
    "string"
  ],
  "slots": []
}
```


Note that the return value of a successful execution also contains basic statistical information (fitness, number of invocations of fitness function for computing fitness values). The synchronous execution of the Heuristic Optimizer is included for experimentation purposes. In a production version of the

SlotMachine system, the Heuristic Optimizer would only be invoked using the asynchronous implementation of the optimization run.

Responses

Status	Meaning	Description	Schema
200	OK	OK	OptimizationResultDTO
401	Unauthorized	Unauthorized	None
403	Forbidden	Forbidden	None
404	Not Found	Not Found	None

`GET /optimizations/{optId}/result`

Get the n *best solutions found by an optimization run, if available.*

Parameters

Name	In	Type	Required	Description
optId	path	string	true	the optimization's identifier
limit	query	integer(int32)	false	the number of solutions to be returned

Example responses

200 Response

```
[
  {
    "fitness": 0,
    "fitnessFunctionInvocations": 0,
    "optId": "string",
    "optimizedFlightSequence": [
      "string"
    ],
    "slots": []
  }
]
```

Responses

Status	Meaning	Description	Schema
200	OK	OK	[OptimizationResultDTO]
401	Unauthorized	Unauthorized	None

Status	Meaning	Description	Schema
403	Forbidden	Forbidden	None
404	Not Found	Not Found; no result available or optimization does not exist	None

`GET /optimizations/{optId}/stats`

Get current statistics for a specific optimization and its results, if available.

Parameters

Name	In	Type	Required	Description
optId	path	string	true	the optimization's identifier

Example responses

200 Response


```
{
  "duration": {
    "nano": 0,
    "negative": true,
    "seconds": 0,
    "units": [
      {
        "dateBased": true,
        "duration": {},
        "durationEstimated": true,
        "timeBased": true
      }
    ],
    "zero": true
  },
  "initialFitness": 0,
  "iterations": 0,
  "optId": "string",
  "requestTime": "2019-08-24T14:15:22Z",
  "resultFitness": 0,
  "status": "CREATED",
  "timeAborted": "2019-08-24T14:15:22Z",
  "timeCreated": "2019-08-24T14:15:22Z",
  "timeFinished": "2019-08-24T14:15:22Z",
  "timeStarted": "2019-08-24T14:15:22Z"
}
```

Note that the schema of the captured statistics may be revised during the work on the Dashboard component; the statistics obtained from the Heuristic Optimizer are a data source for the dashboard. Additional measures may be added, other measures may be dropped.

Responses

Status	Meaning	Description	Schema
200	OK	OK	OptimizationStatisticsDTO
401	Unauthorized	Unauthorized	None
403	Forbidden	Forbidden	None
404	Not Found	Not Found	None

Schemas

In the following, we describe the schema of the data transfer objects.

SlotDTO

```
{
  "time": "2019-08-24T14:15:22Z"
}
```

Name	Type	Required	Restrictions	Description
time	string(date-time)	false	none	none

OptimizationDTO

```
{
  "fitnessEstimator": "string",
  "flights": [
    {
      "flightId": "string",
      "margins": {
        "scheduledTime": "2019-08-24T14:15:22Z",
        "timeNotAfter": "2019-08-24T14:15:22Z",
        "timeNotBefore": "2019-08-24T14:15:22Z",
        "timeWished": "2019-08-24T14:15:22Z"
      },
      "scheduledTime": "2019-08-24T14:15:22Z",
      "weightMap": [
        0
      ]
    }
```

```
  ],
  "initialFlightSequence": [
    "string"
  ],
  "optId": "string",
  "optimizationFramework": "string",
  "optimizationMode": "PRIVACY_PRESERVING",
  "optimizationStatus": "CREATED",
  "parameters": {},
  "privacyEngineEndpoint": "string",
  "slots": [
    {
      "time": "2019-08-24T14:15:22Z"
    }
  ],
  "timestamp": "2019-08-24T14:15:22Z"
}
```


Name	Type	Required	Restrictions	Description
fitnessEstimator	string	false	none	none
flights	[FlightDTO]	false	none	none
initialFlightSequence	[string]	false	none	none
optId	string	false	none	none
optimizationFramework	string	false	none	none
optimizationMode	string	false	none	none
optimizationStatus	string	false	none	none
parameters	object	false	none	none
privacyEngineEndpoint	string	false	none	none
slots	[SlotDTO]	false	none	none
timestamp	string(date-time)	false	none	none

Property	Value
optimizationMode	PRIVACY_PRESERVING
optimizationMode	NON_PRIVACY_PRESERVING
optimizationStatus	CREATED
optimizationStatus	INITIALIZED
optimizationStatus	RUNNING
optimizationStatus	CANCELLED
optimizationStatus	DONE

OptimizationStatisticsDTO


```
{
  "duration": {
    "nano": 0,
    "negative": true,
    "seconds": 0,
    "units": [
      {
        "dateBased": true,
        "duration": {},
        "durationEstimated": true,
        "timeBased": true
      }
    ],
    "zero": true
  },
  "initialFitness": 0,
  "iterations": 0,
  "optId": "string",
  "requestTime": "2019-08-24T14:15:22Z",
  "resultFitness": 0,
  "status": "CREATED",
  "timeAborted": "2019-08-24T14:15:22Z",
  "timeCreated": "2019-08-24T14:15:22Z",
  "timeFinished": "2019-08-24T14:15:22Z",
  "timeStarted": "2019-08-24T14:15:22Z"
}
```


Name	Type	Required	Restrictions	Description
duration	Duration	false	none	none
initialFitness	number(double)	false	none	none
iterations	integer(int32)	false	none	none
optId	string	false	none	none
requestTime	string(date-time)	false	none	none
resultFitness	number(double)	false	none	none
status	string	false	none	none
timeAborted	string(date-time)	false	none	none
timeCreated	string(date-time)	false	none	none
timeFinished	string(date-time)	false	none	none
timeStarted	string(date-time)	false	none	none

Property	Value
status	CREATED

Property	Value
status	IN_PROGRESS
status	FINISHED
status	ABORTED

TemporalUnit


```
{
  "dateBased": true,
  "duration": {
    "nano": 0,
    "negative": true,
    "seconds": 0,
    "units": [
      {
        "dateBased": true,
        "duration": {},
        "durationEstimated": true,
        "timeBased": true
      }
    ],
    "zero": true
  },
  "durationEstimated": true,
  "timeBased": true
}
```


Name	Type	Required	Restrictions	Description
dateBased	boolean	false	none	none
duration	Duration	false	none	none
durationEstimated	boolean	false	none	none
timeBased	boolean	false	none	none

MarginsDTO


```
{
  "scheduledTime": "2019-08-24T14:15:22Z",
  "timeNotAfter": "2019-08-24T14:15:22Z",
  "timeNotBefore": "2019-08-24T14:15:22Z",
  "timeWished": "2019-08-24T14:15:22Z"
}
```

Name	Type	Required	Restrictions	Description
scheduledTime	string(date-time)	false	none	none
timeNotAfter	string(date-time)	false	none	none
timeNotBefore	string(date-time)	false	none	none
timeWished	string(date-time)	false	none	none

FlightDTO


```
{
  "flightId": "string",
  "margins": {
    "scheduledTime": "2019-08-24T14:15:22Z",
    "timeNotAfter": "2019-08-24T14:15:22Z",
    "timeNotBefore": "2019-08-24T14:15:22Z",
    "timeWished": "2019-08-24T14:15:22Z"
  },
  "scheduledTime": "2019-08-24T14:15:22Z",
  "weightMap": [
    0
  ]
}
```


Name	Type	Required	Restrictions	Description
flightId	string	false	none	none
margins	MarginsDTO	false	none	none
scheduledTime	string(date-time)	false	none	none
weightMap	[integer]	false	none	none

Duration


```
{
  "nano": 0,
  "negative": true,
  "seconds": 0,
  "units": [
    {
      "dateBased": true,
      "duration": {
        "nano": 0,
        "negative": true,
        "seconds": 0,
        "units": [],
        "zero": true
```


```
      },
      "durationEstimated": true,
      "timeBased": true
    }
  ],
  "zero": true
}
```


Name	Type	Required	Restrictions	Description
nano	integer(int32)	false	none	none
negative	boolean	false	none	none
seconds	integer(int64)	false	none	none
units	[TemporalUnit]	false	none	none
zero	boolean	false	none	none

OptimizationResultDTO


```
{
  "fitness": 0,
  "fitnessFunctionInvocations": 0,
  "optId": "string",
  "optimizedFlightSequence": [
    "string"
  ],
  "slots": []
}
```


Name	Type	Required	Restrictions	Description
fitness	number(double)	false	none	none
fitnessFunctionInvocations	integer(int32)	false	none	none
optId	string	false	none	none
optimizedFlightSequence	[string]	false	none	none
slots	[LocalDateTime]	false	none	none

4.3 Network Management Function

The Network Management Function (NMF) provides information about active regulations and the current flight list for a given airport. Additionally, this service takes care of validating inputs from SlotMachine on the network level. The SlotMachine Controller component submits solutions to the flight prioritization problem as proposed new flight lists to the NMF and receives – if any of the

proposed new flight lists is accepted by NMF – confirmation about which candidate flight list becomes the actual new flight list.

GET /active

Check if container is active

Example responses

200 Response

```
{
  "active": true,
  "auth": false,
  "repos": false,
  "watermark": false,
  "billing": false,
  "scopes": [
    "admin",
    "write",
    "read"
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	OK	success	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
» active	boolean	false	none	none
» auth	boolean	false	none	none
» repos	boolean	false	none	none
» watermark	boolean	false	none	none
» billing	boolean	false	none	none
» scopes	[string]	false	none	none

GET /info

container overview

Example responses

200 Response

```
{
  "uid": "string",
  "title": "string",
  "image": "string",
  "records": 0
}
```


Responses

Status	Meaning	Description	Schema
200	OK	success	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
» uid	string	false	none	none
» title	string	false	none	none
» image	string	false	none	none
» records	integer	true	none	none

POST /FlightListUpload/{filename}

Upload a flight list response in XML format

Body parameter

```
file: string
```

Parameters

Name	In	Type	Required	Description
filename	path	string	true	none
body	body	object	false	none
» file	body	string(binary)	false	none

Example responses

200 Response


```
<?xml version="1.0" encoding="UTF-8" ?>
<FlightList>
  <content>string</content>
</FlightList>
```


Responses

Status	Meaning	Description	Schema
200	OK	success	FlightList
500	Internal Server Error	server error	None

POST /flightListByAerodrome

configure response to GET /flightListByAerodrome

Parameters

Name	In	Type	Required	Description
identifier	query	string	true	none

Responses

Status	Meaning	Description	Schema
200	OK	success	None
500	Internal Server Error	server error	None

GET /flightListByAerodrome

Get current flight list for an aerodrome

Parameters

Name	In	Type	Required	Description
aerodrome	query	string	false	aerodrome id (ICAO code)

Example responses

200 Response

```
<?xml version="1.0" encoding="UTF-8" ?>
<FlightList>
  <content>string</content>
</FlightList>
```

Responses

Status	Meaning	Description	Schema
200	OK	success	FlightList
500	Internal Server Error	server error	None

POST /flightProposals

Provide list of flight proposals

Body parameter

```
{
  "controller": "string",
  "regulation": "string",
  "optimizaions": [
    {
      "id": "string",
      "prio": 0,
      "flights": [
        {
          "flight": "string",
          "proposed": 0
        }
      ]
    }
  ]
}
```

Parameters

Name	In	Type	Required	Description
body	body	FlightProposal	false	none

Responses

Status	Meaning	Description	Schema
200	OK	success	None
500	Internal Server Error	server error	None

POST /selectedFlightList

Configure response to POST /flightProposals

Body parameter

```
{
  "endpoint": "http://controller:8090",
  "optimization_id": "string",
  "solution_id": "string"
}
```

Parameters

Name	In	Type	Required	Description
body	body	object	false	
» endpoint	body	string	true	
» optimization_id	body	string	true	
» solution_id	body	string	true	

Responses

Status	Meaning	Description	Schema
200	OK	success	None
500	Internal Server Error	server error	None

Schemas

In the following, we describe the schema of the data transfer objects.

FlightList

```
{
  "content": "string"
}
```

Name	Type	Required	Description
content	string	false	

FlightProposal

```
{
  "controller": "string",
  "regulation": "string",
  "optimizaions": [
    {
      "id": "string",
      "prio": 0,
      "flights": [
        {
          "flight": "string",
          "proposed": 0
        }
      ]
    }
  ]
}
```

Name	Type	Required	Description
controller	string	false	unique string identifying SlotMachine Controller
regulation	string	false	name of regulation
optimizaions	[object]	false	list of optimization proposals
» id	string	false	some text
» prio	integer	false	priority
» flights	[object]	false	list of flights in this proposal
»» flight	string	false	flight number
»» proposed	integer	false	timestamp

4.4 Airspace User

The Airspace User component consists of a REST interface (Section 4.4.1) for interaction with the Controller and a WebSocket interface (Section 4.4.2) for notifying the AU web app about regulations and found/accepted solutions.

4.4.1 REST Interface

The AU REST interface is the main component for the AU component to communicate with the SlotMachine system. AU REST handles authentication and authorization and all types of communication between AUs and the SlotMachine system.

All requests are checked against a login/token service, i.e., if the user is still logged in and allowed to send the request.

`POST /AURESTService/rejectSolutions`

Reject solution(s) provided by the optimization run.

Body parameter

```
{
  "airlineId": "SWISS",
  "regulationId": "Reg1234",
  "optimizationId": "Opt5690",
  "rejectedSolutions": [
    {
      "airlineId": "SWISS",
      "regulationId": "Reg1234",
      "optimizationId": "Opt5690",
      "solutionId": "1"
    }
```


```
    ]
}
```

Parameters

Name	In	Type	Required	Description
body	body	RejectedSolutionListDTO	true	Reject solution(s) provided by the optimization run.

Responses

Status	Meaning	Description	Schema
200	OK	Resource created	string
400	Bad Request	Bad Request	string
500	Internal Server Error	Error during registration!	string
default	Default	The rejection	string

`POST /AURESTService/registerForRegulation`

Register an airspace user (airline) for a regulation type on an airport.

Parameters

Name	In	Type	Required	Description
airline	query	string	true	The airline identifier
airport	query	string	true	The airport identifier
regulationType	query	string	true	The regulation type

Responses

Status	Meaning	Description	Schema
200	OK	Resource created	string
400	Bad Request	Bad Request	string
500	Internal Server Error	Error during registration!	string
default	Default	The registration	string

POST /AURESTService/pushOptimizationFlightList

Send the airspace user's preferences for each flight that should be part of the optimization.

Body parameter

```
{
  "requestId": "string",
  "optimizationSession": "string",
  "flights": [
    {
      "flightId": "string",
      "scheduledTime": "2019-08-24T14:15:22Z",
      "priority": 0,
      "notBefore": "2019-08-24T14:15:22Z",
      "notAfter": "2019-08-24T14:15:22Z",
      "timeWished": "2019-08-24T14:15:22Z"
    }
  ]
}
```

Parameters

Name	In	Type	Required	Description
body	body	AusPrefFlightList	true	The airspace user's preferences for each flight that should be part of the optimization.

Example responses

200 Response

```
{
  "regulationId": "string",
  "optimizationId": "0d6b6665-5948-4218-bf74-bb86e549040e",
  "airlineId": "SWR",
  "flights": [
    {
      "flightId": "F01",
      "weightMap": [
        {
          "slotTime": "2019-08-24T14:15:00Z",
          "weight": "1000"
        },
        {
          "slotTime": "2019-08-24T14:20:00Z",
          "weight": "1200"
        },
      ]
    }
```


```
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Resource created	WeightMapDTO
400	Bad Request	Bad Request	string
500	Internal Server Error	Error during registration!	WeightMapDTO
default	Default	The registration	string

`POST /AURESTService/invokeRegulationNotification`

A testing service for sending a new regulation notification to the connected Controller via REST interface.

Parameters

Name	Name	Name	In	Type
airline	query	string	true	The airline identifier
airport	query	string	true	The airport identifer
regulationType	query	string	true	The regulation type: DEPARTURE; ARRIVAL, or EN_ROUTE
regulationId	query	string	true	The regulation identifier

Example responses

200 Response

Responses

Status	Meaning	Description	Schema
200	OK	Resource created	string
400	Bad Request	Bad Request	string
500	Internal Server Error	Error during sending!	string
default	Default	The status.	string

POST /AURESTService/invokeNewRegulation

A testing service for triggering a new regulation notification being sent to the web client on the open WebSocket connection.

Parameters

Name	In	Type	Required	Description
airline	query	string	true	The airline identifier
airport	query	string	true	The airport identifier
regulationType	query	string	true	The regulation type: DEPARTURE, ARRIVAL, or EN_ROUTE
regulationId	query	string	true	The regulation identifier

Responses

Status	Meaning	Description	Schema
200	OK	Resource created	string
400	Bad Request	Bad Request	string
500	Internal Server Error	Error during sending	string
default	Default	The status.	string

POST /AURESTService/callbackEndpoint

The notification endpoint for getting notified by the Controller.

Body parameter

```
"{...}"
```

The body may be different types of JSON documents notifying the AU of a change. The body can be a RegulationNotificationDTO or a SolutionNotificationDTO.

Parameters

Name	In	Type	Required	Description
body	body	string	true	The notification objects as string representations of JSON documents.

Responses

Status	Meaning	Description	Schema
200	OK	Resource created	string
400	Bad Request	Bad Request	string
500	Internal Server Error	Error during registration	string
default	Default	The status.	string

`GET /AURESTService/token`

Endpoint for checking if the authentication token for an airspace user is valid.

Parameters

Name	In	Type	Required	Description
authorization	header	string	true	The authentication token
userName	query	string	true	The user name

Responses

Status	Meaning	Description	Schema
200	OK	Token valid	string
400	Bad Request	Bad Request	string
403	Forbidden	Token invalid!	string
default	Default	The auth token.	string

`GET /AURESTService/getFlightListForOptimization`

Get the current flight list for an airline on an airport that can be optimized.

Parameters

Name	In	Type	Required	Description
airlineId	query	string	true	The airline identifier
airportId	query	string	true	The airport identifier

Responses

Status	Meaning	Description	Schema
200	OK	Resource created	FlightListDTO

Status	Meaning	Description	Schema
400	Bad Request	Bad Request	string
500	Internal Server Error	Error during registration!	FlightListDTO
default	Default	The FlightList	FlightListDTO

GET /AURESTService/authenticate

Endpoint authenication and authorization of an airspace user by providing a basic authorization header.

Parameters

Name	In	Type	Required	Description
authorization	header	string	true	Base64 authentication

Responses

Status	Meaning	Description	Schema
200	OK	Token created	string
400	Bad Request	Bad Request	string
403	Forbidden	Login forbidden	string
default	Default	The auth token.	string

GET /AURESTService/actualProvidedSolutions

Get the actual provided solutions for the current optimization run.

Parameters

Name	In	Type	Required	Description
optId	query	string	true	The identifier of the optimization run

Responses

Status	Meaning	Description	Schema
200	OK	Solution(s) provided	SolutionListDTO
400	Bad Request	Bad Request	string
500	Internal Server Error	Error during registration	SolutionListDTO

Status	Meaning	Description	Schema
default	Default	The actual provided Solutions	SolutionListDTO

GET /AURESTService/actualAcceptedSolution

Get the actual accepted solutions for the current optimization session.

Parameters

Name	In	Type	Required	Description
optId	query	string	true	The identifier of the optimization run

Responses

Status	Meaning	Description	Schema
200	OK	Solution(s) accepted!	SolutionListDTO
400	Bad Request	Bad Request	string
500	Internal Server Error	Error during registration!	SolutionListDTO
default	Default	The actual provided Solutions	SolutionListDTO

DELETE /AURESTService/unregisterFromRegulation

Unregister an airspace user (airline) for a regulation type at an airport.

Parameters

Name	In	Type	Required	Description
airline	query	string	true	The airline identifier
airport	query	string	true	The airport identifier
regulationType	query	string	true	The type of regulation: DEPARTURE, ARRIVAL, or EN_ROUTE

Responses

Status	Meaning	Description	Schema
200	OK	Resource created	string
400	Bad Request	Bad Request	string

Status	Meaning	Description	Schema
500	Internal Server Error	Error during registration!	string
default	Default	The unregistration	string

Schemas

In the following, we describe the schema of the data transfer objects that are exchanged between the Airspace User REST interface and other components.

RegulationNotficationDTO

```
{
  "airlineId": "string",
  "airportId": "string",
  "regulationId": "string
}
```

SolutionNotficationDTO

```
{
  "optimizationId": "string”,
  "regulationId": "string”,
  "airportId": "string",
  "airlineId": "string",
  "airportId": "string",
  "solutionStatus": "string
}
```

AusPrefFlightList

```
{
  "requestId": "string",
  "optimizationSession": "string",
  "flights": [
    {
      "flightId": "string",
      "scheduledTime": "2019-08-24T14:15:22Z",
      "priority": 0,
      "notBefore": "2019-08-24T14:15:22Z",
      "notAfter": "2019-08-24T14:15:22Z",
      "timeWished": "2019-08-24T14:15:22Z"
    }
  ]
}
```

Name	Name	Name	Type	Required
requestId	string	false	none	The unique request ID from getFlightListForOptimization
optimizationSession	string	false	none	the optimization's identifier
flights	[PrefFlight]	false	none	The list of Flights (FlightDTO)

PrefFlight

```
{
  "flightId": "string",
  "scheduledTime": "2019-08-24T14:15:22Z",
  "priority": 0,
  "notBefore": "2019-08-24T14:15:22Z",
  "notAfter": "2019-08-24T14:15:22Z",
  "timeWished": "2019-08-24T14:15:22Z"
}
```

Name	Type	Required	Description
flightId	string	false	The unique Flight identification
scheduledTime	string(date-time)	false	The slot time of the flight
priority	integer(int32)	false	a numeric priority
notBefore	string(date-time)	false	The UTC time before the flight should not be scheduled
notAfter	string(date-time)	false	The UTC time after that the flight should not be scheduled
timeWished	string(date-time)	false	The preferred UTC time of the flight schedule in

WeightMapDTO

```
{
  "regulationId": "string",
  "optimizationId": "string",
  "airlineId": "string",
  "flights": [
    {
      "flightId": "string",
      "weightMap": [
        {
          "slotTime": "2019-08-24T14:15:22Z",
          "weight": "string"
        }
      ]
    }
```


```
  ]
}
```

Name	Type	Required	Description
regulationId	string	false	the regulation identifier
optimizationId	string	false	the optimization's identifier
airlineId	string	false	The airline identifier
flights	[WeightMapFlightDTO]	false	The weight map for each flight and slot combination

WeightMapFlightDTO

```
{
  "flightId": "string",
  "weightMap": [
    {
      "slotTime": "2019-08-24T14:15:22Z",
      "weight": "string"
    }
  ]
}
```

Name	Type	Required	Description
flightId	string	false	The flight identifier
weightMap	[WeightMapObjectDTO]	false	A weight map assigning a weight/utility to each slot

WeightMapObjectDTO

```
{
  "slotTime": "2019-08-24T14:15:22Z",
  "weight": "string"
}
```

Name	Type	Required	Description
slotTime	string(date-time)	false	The time and date of the slot.
weight	string	false	The assigned weight

Flight

```
{
  "flightId": "string",
  "keys": {
    "aircraftId": "string",
```

```
    "aerodromeOfDeparture": "string",
    "nonICAOAerodromeOfDeparture": true,
    "airFiled": true,
    "aerodromeOfDestination": "string",
    "nonICAOAerodromeOfDestination": true,
    "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
  },
  "aircraftType": "string",
  "scheduledTakeOffTime": "2019-08-24T14:15:22Z",
  "estimatedTakeOffTime": "2019-08-24T14:15:22Z",
  "aircraftOperator": "string",
  "operatingAircraftOperator": "string",
  "slotIssued": true,
  "delay": 0,
  "mostPenalisingRegulation": "string",
  "filedRegistrationMark": "string",
  "slotSwapCounter": {
    "currentCounter": 0,
    "maxLimit": 0
  }
}
```


Name	Type	Required	Description
flightId	string	true	The flight identifier
keys	FlightKey	false	
aircraftType	string	false	
scheduledTakeOffTime	string(date-time)	false	
estimatedTakeOffTime	string(date-time)	false	
aircraftOperator	string	false	
operatingAircraftOperator	string	false	
slotIssued	boolean	false	
delay	integer(int32)	false	
mostPenalisingRegulation	string	false	
filedRegistrationMark	string	false	
slotSwapCounter	SlotSwapCounter	false	

FlightKey


```
{
  "aircraftId": "SWR110",
  "aerodromeOfDeparture": "FAOR",
  "nonICAOAerodromeOfDeparture": false,
  "airFiled": false,
  "aerodromeOfDestination": "LOWW",
  "nonICAOAerodromeOfDestination": true,
```

```
  "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
}
```


Name	Type	Required	Description
aircraftId	string	true	The aircraft identifier
aerodromeOfDeparture	string	true	The (ICAO) identifier of departure aerodrome
nonICAOAerodromeOfDeparture	boolean	false	Whether the departure aerodrome is not recognized by ICAO
airFiled	boolean	false	
aerodromeOfDestination	string	true	The (ICAO) identifier of destination aerodrome
nonICAOAerodromeOfDestination	boolean	false	Whether the destination aerodrome is not recognized by ICAO
estimatedOffBlockTime	string(date-time)	false	Estimated time that the aircraft starts movement associated with departure

FlightList


```
{
  "requestReceptionTime": "2019-08-24T14:15:22Z",
  "requestId": "string",
  "optimizationSession": "string",
  "regulationId": "string",
  "currentServerTime": "2019-08-24T14:15:22Z",
  "cutOffTime": "2019-08-24T14:15:22Z",
  "nextOptRun": "2019-08-24T14:15:22Z",
  "slots": [
    {
      "slotTime": "2019-08-24T14:15:22Z"
    }
  ],
  "flights": [
    {
      "flightId": "string",
      "keys": {
        "aircraftId": "string",
        "aerodromeOfDeparture": "string",
        "nonICAOAerodromeOfDeparture": true,
        "airFiled": true,
        "aerodromeOfDestination": "string",
        "nonICAOAerodromeOfDestination": true,
        "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
      },
      "aircraftType": "string",
```

```
      "scheduledTakeOffTime": "2019-08-24T14:15:22Z",
      "estimatedTakeOffTime": "2019-08-24T14:15:22Z",
      "aircraftOperator": "string",
      "operatingAircraftOperator": "string",
      "slotIssued": true,
      "delay": 0,
      "mostPenalisingRegulation": "string",
      "filedRegistrationMark": "string",
      "slotSwapCounter": {
        "currentCounter": 0,
        "maxLimit": 0
      }
    }
  ]
}
```


Name	Type	Name	Required
requestReceptionTime	string(date-time)	false	The time when the getFlightList request has been answered
requestId	string	false	The unique request identifier
optimizationSession	string	false	the optimization's identifier
regulationId	string	false	the regulation identifier
currentServerTime	string(date-time)	false	current time on the SlotMachine instance
cutOffTime	string(date-time)	false	latest time the AU can send his preferences
nextOptRun	string(date-time)	false	next time the optimization run is scheduled
slots	[Slot]	false	the available slots for the next optimization run
flights	[Flight]	false	list of flights that can be part of the optimization

Slot


```
{
  "slotTime": "2019-08-24T14:15:22Z"
}
```


Name	Type	Required	Description
slotTime	string(date-time)	false	The date and time of the slot

SlotSwapCounter


```
{
  "currentCounter": 0,
  "maxLimit": 0
}
```


Name	Type	Name	Description
currentCounter	integer(int32)	false	
maxLimit	integer(int32)	false	

FlightDTO


```
{
  “flightId”: “string”,
  “keys”: {
    “aircraftId”: “string”,
    “aerodromeOfDeparture”: “string”,
    “nonICAOAerodromeOfDeparture”: true,
    “airFiled”: true,
    “aerodromeOfDestination”: “string”,
    “nonICAOAerodromeOfDestination”: true,
    “estimatedOffBlockTime”: “2019-08-24T14:15:22Z”
  },
  “aircraftType”: “string”,
  “scheduledTakeOffTime”: “2019-08-24T14:15:22Z”,
  “estimatedTakeOffTime”: “2019-08-24T14:15:22Z”,
  “aircraftOperator”: “string”,
  “operatingAircraftOperator”: “string”,
  “slotIssued”: true,
  “delay”: 0,
  “mostPenalisingRegulation”: “string”,
  “filedRegistrationMark”: “string”,
  “slotSwapCounter”: {
    “currentCounter”: 0,
    “maxLimit": 0
  }
}
```


Name	Type	Required	Description
flightId	string	false	The flight identifier
keys	FlightKeyDTO	false	
aircraftType	string	false	
scheduledTakeOffTime	string(date-time)	false	
estimatedTakeOffTime	string(date-time)	false	
aircraftOperator	string	false	
operatingAircraftOperator	string	false	

slotIssued	boolean	false
delay	integer(int32)	false
mostPenalisingRegulation	string	false
filedRegistrationMark	string	false
slotSwapCounter	SlotSwapCounterDTO	false

FlightKeyDTO


```
{
  "aircraftId": "string",
  "aerodromeOfDeparture": "string",
  "nonICAOAerodromeOfDeparture": true,
  "airFiled": true,
  "aerodromeOfDestination": "string",
  "nonICAOAerodromeOfDestination": true,
  "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
}
```


Name	Type	Required	Description
aircraftId	string	false	The aircraft identifier
aerodromeOfDeparture	string	false	
nonICAOAerodromeOfDeparture	boolean	false	
airFiled	boolean	false	
aerodromeOfDestination	string	false	
nonICAOAerodromeOfDestination	boolean	false	
estimatedOffBlockTime	string(date-time)	false	

FlightListDTO


```
{
  "optimization": {
    "regulationId": "LDCTB04D",
    "requestReceptionTime": "2019-08-24T14:15:22Z",
    "requestId": "B2B_CUR:45318",
    "optimizationId": "string",
    "state": "WAIT_FOR_INPUTS",
    "currentServerTime": "2019-08-24T14:15:22Z",
    "cutOffTime": "2019-08-24T14:15:22Z",
    "nextOptimizationRun": "2019-08-24T14:15:22Z"
  },
  "slots": [
```

```
    {
      "slotTime": "2019-08-24T14:15:22Z"
    }
  ],
  "flights": [
    {
      "flightId": "F1",
      "keys": {
        "aircraftId": "SWR139",
        "aerodromeOfDeparture": "LOWW",
        "nonICAOAerodromeOfDeparture": false,
        "airFiled": true,
        "aerodromeOfDestination": "LSZH",
        "nonICAOAerodromeOfDestination": false,
        "estimatedOffBlockTime": "2019-08-24T14:15:22Z"
      },
      "aircraftType": "A333",
      "scheduledTakeOffTime": "2019-08-24T14:15:22Z",
      "estimatedTakeOffTime": "2019-08-24T14:15:22Z",
      "aircraftOperator": "SWR",
      "operatingAircraftOperator": "SWR",
      "slotIssued": true,
      "delay": 0,
      "mostPenalisingRegulation": "string",
      "filedRegistrationMark": "string",
      "slotSwapCounter": {
        "currentCounter": 0,
        "maxLimit": 0
      }
    }
  ]
}
```


Name	Type	Required	Description
optimization	OptimizationDTO	false	The optimization information
slots	[SlotDTO]	false	The available slots for the optimization
flights	[FlightDTO]	false	The flights for the optimization

OptimizationDTO


```
{
  "regulationId": "string",
  "requestReceptionTime": "2019-08-24T14:15:22Z",
  "requestId": "string",
  "optimizationId": "string",
  "state": "WAIT_FOR_INPUTS",
  "currentServerTime": "2019-08-24T14:15:22Z",
```


```
  "cutOffTime": "2019-08-24T14:15:22Z",
  "nextOptimizationRun": "2019-08-24T14:15:22Z"
}
```


Name	Type	Required	Description
regulationId	string	false	The regulation identifier
requestReceptionTime	string(date-time)	false	The time when the getFlightList request has be answered
requestId	string	false	the unique request id
optimizationId	string	false	the optimization identifier
state	string	false	The state of the optimization: WAIT_FOR_INPUTS, CUT_OFF_TIME_REACHED, OPTIMIZATION_RUNNING, WAITING_FOR_REJECTS, WAITING_FOR_NM_ CONFIRMATION
currentServerTime	string(date-time)	false	the current time on the SlotMachine instance
cutOffTime	string(date-time)	false	latest time the AU can send his preferences
nextOptimizationRun	string(date-time)	false	the time when the next optimization is starting

SlotDTO


```
{
  "slotTime": "2019-08-24T14:15:22Z"
}
```


Name	Type	Required	Description
slotTime	string(date-time)	false	

SlotSwapCounterDTO


```
{
  "currentCounter": 0,
  "maxLimit": 0
}
```

Name	Type	Required	Description
currentCounter	integer(int32)	false	
maxLimit	integer(int32)	false	

SolutionDTO

```
{
  "regulationId": "string",
  "optimizationId": "string",
  "solutionId": "string",
  "priority": 0,
  "flights": [
    {
      "flightId": "string",
      "slotTime": "2019-08-24T14:15:22Z"
    }
  ]
}
```

Name	Type	Required	Description
regulationId	string	false	the regulation identifier
optimizationId	string	false	the optimization identifier
solutionId	string	false	the unique id of the solution
priority	integer(int32)	false	the priority of this solution
flights	[SolutionFlightDTO]	false	flight information for the solution

SolutionFlightDTO

```
{
  "flightId": "string",
  "slotTime": "2019-08-24T14:15:22Z"
}
```

Name	Type	Required	Description
flightId	string	false	The flight identifier
slotTime	string(date-time)	false	The date and time of the slot

SolutionListDTO


```
{
  "regulationId": "string",
  "optimizationId": "string",
  "currentServerTime": "2019-08-24T14:15:22Z",
  "rejectUntil": "2019-08-24T14:15:22Z",
  "solutions": [
    {
      "regulationId": "string",
      "optimizationId": "string",
      "solutionId": "string",
      "priority": 0,
      "flights": [
        {
          "flightId": "string",
          "slotTime": "2019-08-24T14:15:22Z"
        }
      ]
    }
  ]
}
```


Name	Type	Required	Description
regulationId	string	false	the regulation identifier
optimizationId	string	false	the optimization identifier
currentServerTime	string(date-time)	false	current time on the SoltMachine instance
rejectUntil	string(date-time)	false	the time until a reject can be send
solutions	[SolutionDTO]	false	the solutions to be rejected

4.4.2 WebSocket Interface

The WebSocket interface serves the AU REST component to notify the AU web app about issued regulations, proposed new flight lists, and the flight list finally accepted by the NMF. Exchange of heartbeat messages between AU REST and AU web app guarantees a stable connection.

The Websocket Endpoint is reachable via /AUClientRestWSEndpoint on each AU instance.

Heartbeat is initiated by the AU web app by sending a HeartbeatRequest message, and the AU REST component returns a HeartbeatResponse message.

HeartbeatRequest

```
{
  “requestId”:<uuid>,
  “mutation”:“HB_REQUEST”,
  “sendTime”:<UTCDateTime>
}
```

HeartbeatResponse

```
{
  “requestId”:<uuid>,
  “mutation”:“HB_RESPONSE”,
  “sendTime”:<UTCDateTime>,
  “state”:<CONNECTED>
}
```

In the following, we define the messages sent from the AU REST component to the AU web app.

NewRegulationNotification:

```
{
  “mutation”:“NEW_REGULATION_AVAILABLE”,
  “airspaceUser”:<aispaceUserId>
  “airport”:<airport>,
  “regulationId”:<regulationId>
}
```

NewProposedSolutionsNotification:

```
{
  “mutation”:“NEW_PROPOSED_SOLUTIONS_AVAILABLE”,
  “optimizationSessionId”:<optimiziationSessionId>
}
```

NewAcceptedSolutionsNotification:

```
{
  “mutation”:“NEW_ACCEPTED_SOLUTION_AVAILABLE”,
  “optimizationSessionId”:<optimiziationSessionId>
}
```

4.5 Dashboard

We do not intend to implement a web service interface for accessing the Dashboard component. Rather, the Dashboard component is a user interface that retrieves different metric data to be displayed from the interfaces provided by other components, particularly the Heuristic Optimizer's returned statistics about optimizations. The main component to provide aggregated data for the dashboard, however, will be the Controller. The definition of those interfaces and interaction will depend on the further development and will be specified and finalized at a later point.

4.6 Privacy Engine and MPC Nodes

PUT /sessionClear

Create a new session using a clear-text weight-map. The weight-map is a rectangular matrix (list of lists)

Body parameter

```
[[10,0,0],[0,8,4],[0,3,9]]
```

Parameters

Name	In	Type	Required	Description
body	body	array[array]	false	none

Responses

Status	Meaning	Description	Schema
201	Created	Successful Response	Inline
400	Bad Request	Bad Request	Error
422	Unprocessable Entity	Validation Error	HTTPValidationError
500	Internal Server Error	Internal Server Error	Error

GET /nodes

Get a list of MPC nodes

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline

GET /status

Get the current status of the Privacy Engine.

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
503	Service Unavailable	Service Unavailable	Error

PUT /sessionSecret

Create a new session using secret-shared weight-maps (one for each MPC node). The weight-map is a dictionary, containing for each MPC node ("peer") a rectangular matrix (list of lists). The elements (weights w_{ij}) are encoded (secret shared) for each MPC node and encrypted under the public key of the respective MPC node.

Body parameter

```
{
  "A": [[<<w11 encoded for A> encrypted under pkA>,<<w12 … > … >,< … >],
        [<<w21 encoded for A> encrypted under pkA>,<<w22 … > … >,< … >],
        [<<w31 encoded for A> encrypted under pkA>,<<w32 … > … >,< … >],
  "B": […],
  "C": […]
}
```

Parameters

Name	In	Type	Required	Description
body	body	object	false	none
» **additionalProperties**	body	[array]	false	none

Responses

Status	Meaning	Description	Schema
201	Created	Successful Response	Inline
400	Bad Request	Bad Request	Error
422	Unprocessable Entity	Validation Error	HTTPValidationError
500	Internal Server Error	Internal Server Error	Error

PUT /computeFitnessClear

Return a clear-text fitness value for a given list of indices. For a weight map [[12, 13], [23, 34]], an input of [0, 1] should return 46. [1, 0] should return 36, and any other input should return an error.

Body parameter

```
[0,1]
```

Parameters

Name	In	Type	Required	Description
body	body	array[integer]	false	none

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	integer
400	Bad Request	Bad Request	Error
422	Unprocessable Entity	Validation Error	HTTPValidationError
500	Internal Server Error	Internal Server Error	Error

PUT /computePopulationOrder

Return the maximum fitness value and an ordered list of configurations. For a weight-map [[12, 13], [23, 34]], an input of [[0, 1], [1, 0]] should return a maximum of 46 and a list [0, 1].

Body parameter

```
[0,1]
```

Parameters

Name	In	Type	Required	Description
body	body	array[array]	false	none

Example responses

200 Response


```
{
  "maximum": 0,
  "order": [
    0
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	MaxOrderedResponse
400	Bad Request	Bad Request	Error
422	Unprocessable Entity	Validation Error	HTTPValidationError
500	Internal Server Error	Internal Server Error	Error

PUT /computeClearing

Return the calculated transactions of tokens between flights for a given final sequence.

Body parameter


```
[0,1]
```


Parameters

Name	In	Type	Required	Description
body	body	array[integer]	false	none

Example responses

200 Response


```
{
  "values": [
    0
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	ClearingResponse
400	Bad Request	Bad Request	Error
422	Unprocessable Entity	Validation Error	HTTPValidationError
500	Internal Server Error	Internal Server Error	Error

PUT /

Encode

Secret-share a weight-map Input is a rectangular matrix Output is a map from ‘A’, ‘B’, ‘C’ to shares of the input weight-map

Body parameter

```
[]
```

Parameters

Name	In	Type	Required	Description
body	body	array[array]	false	none

Example responses

200 Response

```
{
  "property1": [
    [
      0
    ]
  ],
  "property2": [
    [
      0
    ]
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
422	Unprocessable Entity	Validation Error	HTTPValidationError

Response Schema

Status Code **200**

Response Encode Put

Name	Type	Required	Restrictions	Description
additionalProperties	[array]	false	none	none

Schemas

In the following, we describe the schema of the data transfer objects.

ClearingResponse

```
{
  "values": [
    0
  ]
}
```

Name	Type	Required	Restrictions	Description
values	[number]	true	none	none

Error

```
{
  "code": 0,
  "message": "string"
}
```

Name	Type	Required	Restrictions	Description
code	integer	false	none	none
message	string	false	none	none

HTTPValidationError

```
{
  "detail": [
    {
      "loc": [
        "string"
      ],
      "msg": "string",
      "type": "string"
    }
```

```
  ]
}
```


Name	Type	Required	Restrictions	Description
detail	[ValidationError]	false	none	none

MaxOrderedResponse


```
{
  "maximum": 0,
  "order": [
    0
  ]
}
```


Name	Type	Required	Restrictions	Description
maximum	integer	false	none	none
order	[integer]	false	none	none

ValidationError


```
{
  "loc": [
    "string"
  ],
  "msg": "string",
  "type": "string"
}
```


Name	Type	Required	Restrictions	Description
loc	[string]	true	none	none
msg	string	true	none	none
type	string	true	none	none

4.7 Blockchain, Credit Wallets, and Wallet Management

The blockchain is used mainly for auditability purposes, to provide transparency for the participating AUs. The AUs' credit balances are periodically pushed to the blockchain in clear text for any AU to see, but not in real-time. In the following, we first define the interfaces for writing to the blockchain before discussing some options for the Credit Wallets and Wallet Management, which at this stage of the development are not yet finally specified.

4.7.1 Blockchain

Each blockchain application provides an interface for basic querying. Furthermore, the blockchain application allows for the creation of a transaction on the blockchain in order to update the credit balances. Only the Credit Management component can update the credit balances since that function is protected and tied to the Credit Management account. The credit balances can be queried by everyone who has access to a blockchain client.

`POST /set_credits`

Sets the current credit balances.

Body parameter

```
[
  {
    "participant": "SWISS",
    "balance": 100
  }
]
```

Parameters

Name	In	Type	Required	Description
body	body	array[object]	true	The specification of the credit balance per participant.

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
400	Bad Request	Bad Request	Error
422	Unprocessable Entity	Validation Error	HTTPValidationError
500	Internal Server Error	Internal Server Error	Error

`GET /credit`

Queries the credit balances for all participants. The output is paginated, i.e., will be returned on multiple page if the result would be long. Pagination options can be set using the parameters.

Parameters

Name	In	Type	Required	Description
pagination.key	query	byte	true	
pagination.offset	query	uint64	true	
pagination.limit	query	uint64	true	
pagination.countTotal	query	boolean	true	The number airspace user balances per page.
pagination.reverse	query	boolean	true	

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
400	Bad Request	Bad Request	Error
422	Unprocessable Entity	Validation Error	HTTPValidationError
500	Internal Server Error	Internal Server Error	Error

`GET /credit/{$participant}`

Queries the credit balance of a particular participant.

Parameters

Name	In	Type	Required	Description
participant	path	string	true	The id of the airspace user that the credit balance should be retrieved for.

Responses

Status	Meaning	Description	Schema
200	OK	Successful Response	Inline
400	Bad Request	Bad Request	Error
422	Unprocessable Entity	Validation Error	HTTPValidationError
500	Internal Server Error	Internal Server Error	Error

4.7.2 Credit Wallets and Wallet Management

The Wallet Management will either be realized as a module (subcomponent) of the Controller or runs in a separate container. In the latter case, the Controller would interact with the Controller via a REST interface. The Credit Wallets could be realized using a database management system, e.g., a relational. In that case, rather than using a REST interface, the Wallet Management could interact with the Credit Wallets using the database management system's update facility, e.g., SQL INSERT/UPDATE statements via JDBC connection.

5 Sequence Diagrams

This chapter defines the data flows between the functional components described in Chapter 3.

5.1 Controller and Network Management Function

The sequence diagram below illustrates the interaction between the Controller component (see Section 3.3) and simulated Network Management Functions (Functional Components section 3.2), as well as a Test Engineer specifying how NMF should response to requests from the Controller.

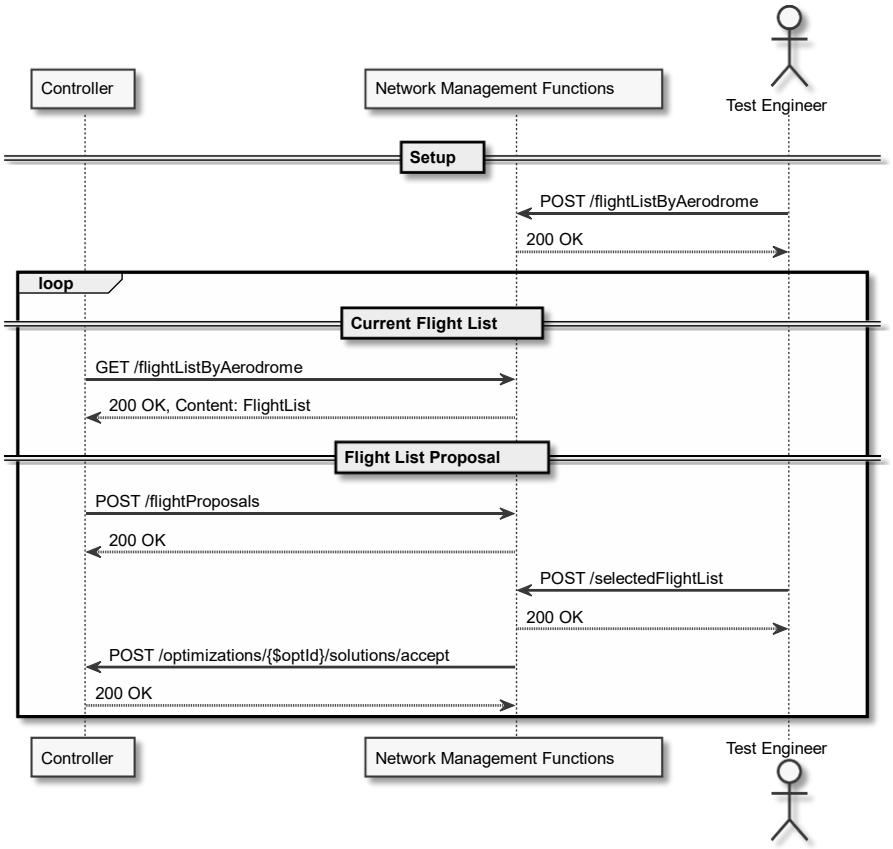

Figure 7. Interaction between Controller and Network Management Functions

5.2 Controller and Airspace User

Figure 8 first illustrates how the AU components are initialized and registered for notifications with the Controller. For informing the AU web app that new information is available, the AU REST component employs a WebSocket connection, which the AU web app initiates. In order to keep the WebSocket connection open, a *heartbeat* is regularly sent from the web app to the REST endpoint, which responds with an acknowledgment. By regularly sending a heartbeat message, the AU web app can constantly check if the connection is still open and sending messages is possible.

Figure 8 then illustrates how the AU REST component is responsible for handling communication between AU and the Controller. For the communication between the AU REST and Controller, the HTTP protocol is used. The AU REST component accesses the Controller's REST endpoint. The AU REST component, in turn, provides a dedicated REST endpoint (*callbackEndpoint*) for the Controller to submit notifications. When the AU subscribes to receiving notifications about airport regulations, the callbackEndpoint is registered with the Controller.

As soon as a regulation is available and the regulation meets the criteria of a registration, i.e., the registration is for the same regulation type and airport, the Controller sends a RegulationNotificationDTO message to the registered AU REST component, which forwards the notification via the WebSocket connection to the AU web app. The web app displays information about the regulation, requesting the flight list for that regulation from the AU REST component, which forwards the request to the Controller. The Controller returns the flight list for the airspace user to the AU REST interface, which forwards the flight list to the web app, which displays the flight list. The AU may then fill in the preferences (TimeWished, NotBefore, NotAfter, Priority).

When all preferences for flights are filled in, an operator from the AU can submit the preferences to the AU REST component, which computes the weights, creating the weight map. In privacy-preserving mode, the weights are encrypted and submitted in encrypted form to the Controller. To this end, the AU REST component uses an encoding service, which can be hosted on the AU's premises (not shown in Figure 8).

As soon as an optimization run has finished, the Controller informs the AU REST component about the availability of proposed solutions (Figure 9). The AU REST component forwards the notification via the open WebSocket connection to the AU web app, which triggers the loading of the proposed flight lists to be displayed on the AU web app so that the AU's operator can see and compare the results. All proposed solutions that do not meet the operator's criteria can be rejected. All solutions that are not rejected will be submitted by the Controller to the NMF for approvement.

When there is a solution accepted by the NMF, the Controller notifies the AU REST component via the WebSocket connection and the AU web app loads the accepted solution from the Controller via the AU REST interface.

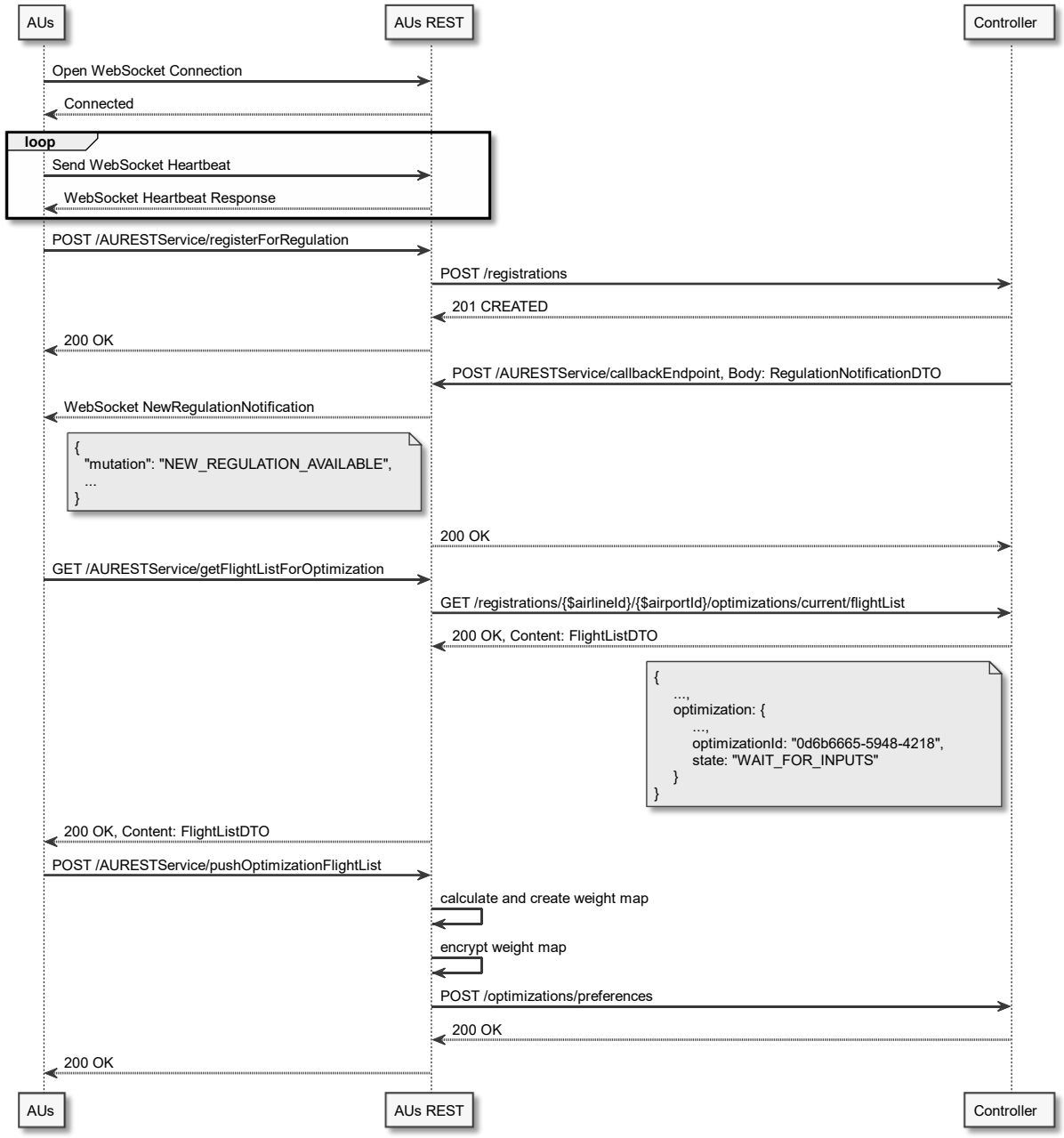

Figure 8. Participation of AU in an optimization run

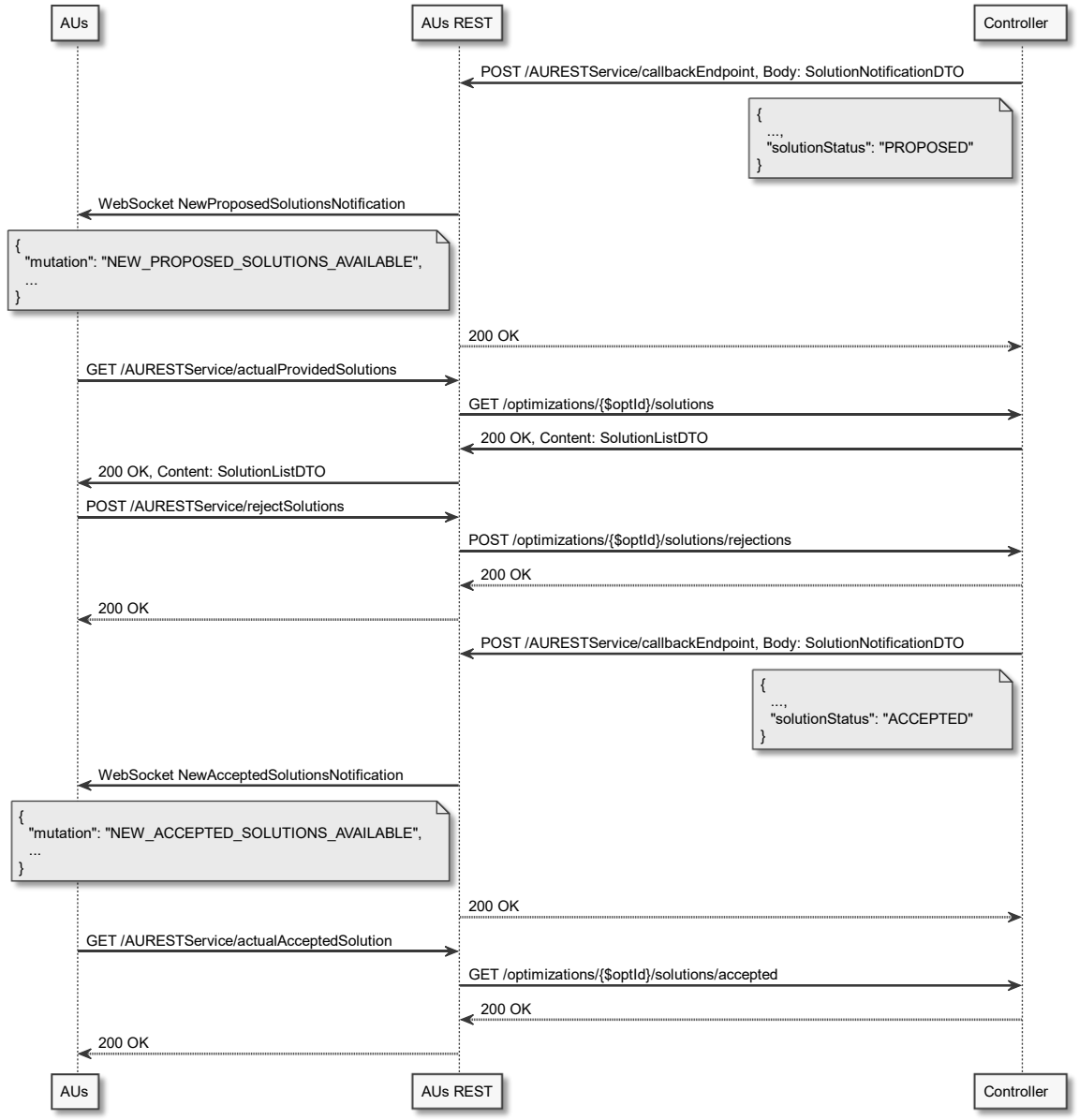

Figure 9. Retrieval of flight list by AU after an optimization run

5.3 Controller, Heuristic Optimizer, Privacy Engine, and MPC Nodes

Figure 10 illustrates the interaction flow of Controller and Heuristic Optimizer on the one hand as well as Privacy Engine and the MPC nodes on the other hand. The Controller sends the encrypted weights submitted by the AUs to the Privacy Engine, which in turn forwards the settings to the different MPC nodes. The Controller then initializes an optimization run in the Heuristic Optimizer and starts the optimization run. The optimization run is an iterative process, the evolutionary algorithm generating in each iteration step a population of flight lists, which are submitted to the Privacy Engine for evaluation. The Privacy Engine submits the data to the MPC nodes for computation, which ultimately produces a ranked list of the input flight lists along with the maximum fitness value in the population. The Heuristic Optimizer employs the feedback from the Privacy Engine to find new solutions. After the specified abort condition is satisfied, the Heuristic Optimizer stops the optimization run. The Heuristic

Optimizer can then obtain the results. There is no notification from Heuristic Optimizer to Controller once the run has finished due to the following reasons. First, the Controller can configure the abortion criteria. Second, the Heuristic Optimizer allows for the abortion of an optimization run and will still return a result. The Heuristic Optimizer also allows to retrieve intermediate results. So the Controller should specify a time limit and after that time has passed should just retrieve the result available at that time. Notification is thus not necessary.

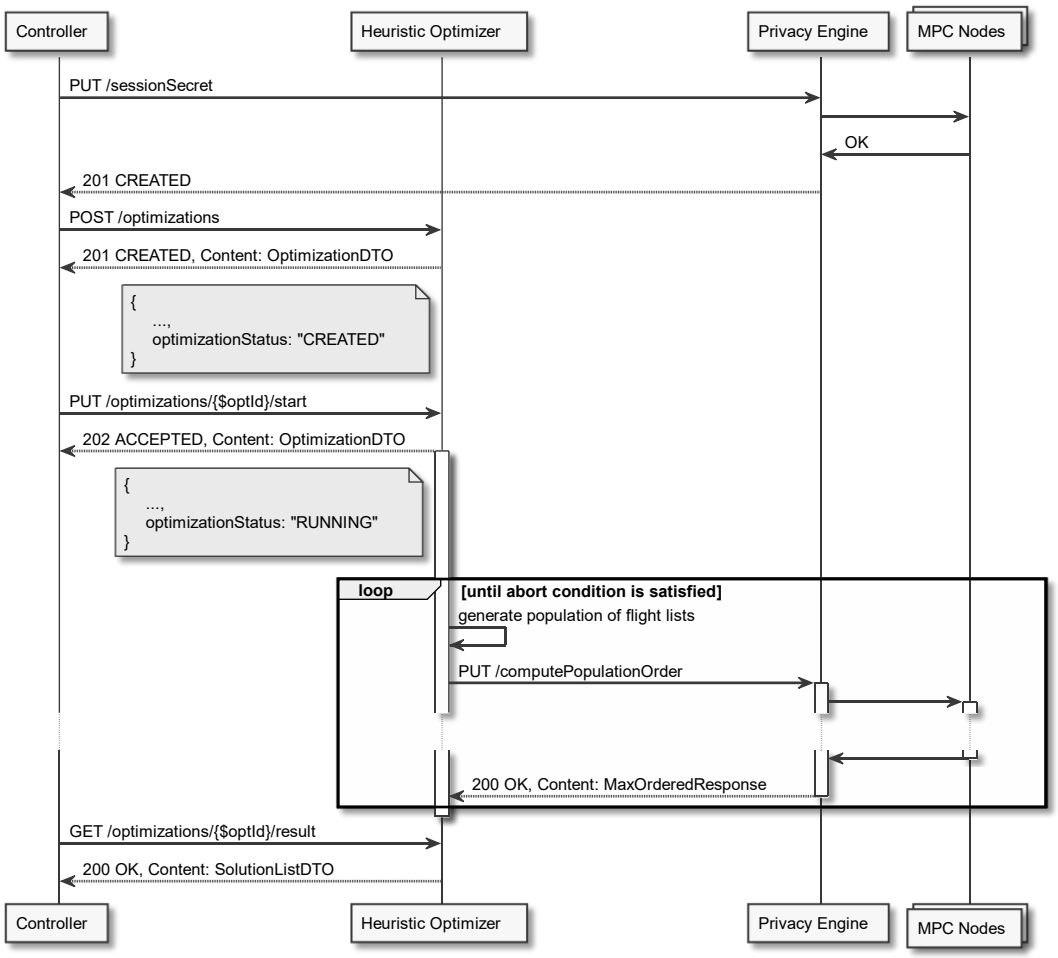

Figure 10. Interaction between Controller, Heuristic Optimizer, Privacy Engine, and MPC nodes when conducting an optimization run

The Privacy Engine will also provide a non-privacy-preserving mode for testing purposes. In this non-privacy-preserving mode, unencrypted weights are submitted as input but the remainder of the interaction flow between Privacy Engine and MPC nodes stays the same.

5.4 Controller, Wallet Management and Blockchain

Different market mechanisms will require different clearing mechanisms, which in turn require different choices regarding the architecture. On the one hand, SM1 defined in D2.3 – Business Concepts [2] does not require a Wallet Management component. However, when realizing SM1, a different clearing mechanism would have to be implemented. On other hand, SM2 and SM3 rely on credits instead of real-world currency. Since airlines are reluctant to participate in market which

involves exchange of real-world currency (see requirement fair_3 in D2.1 [1]), we will focus on a credit-based market mechanism. In the following, we describe the interaction between Controller, Wallet Management, and Blockchain. Note, however, that the following describes the current state of our research, which is still ongoing. The final design may, therefore, be subject to change when we conduct additional experiments and receive stakeholder feedback.

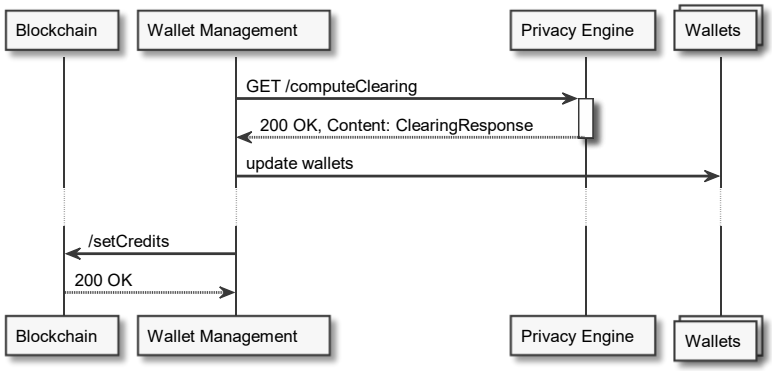

Figure 11. Interaction between Wallet Management, Privacy Engine, Wallets, and Blockchain

Figure 11 shows one proposal for handling the clearing process in the SlotMachine system. The Wallet Management requests the clearing for a certain optimization run from the Privacy Engine. The Privacy Engine computes amounts of credits to be moved between wallets, using the MPC nodes (not shown), based on the previously submitted AU preferences for that optimization run. The Wallet Management component then updates the wallet accordingly; note that the precise interface has not been specified yet and is still subject to discussion. After several optimization runs, the Wallet Management component also pushes the credit balances to the blockchain, in clear text, so that each AU may also view the credit balance of other AUs, in order to increase trust in the fairness of the system.

We also consider other proposals for incorporating the blockchain. For example, zero-knowledge proofs could be incorporated, with the Privacy Engine pushing directly to the blockchain in order to make the system auditable in real-time. Wallet Management could also be included in the Privacy Engine altogether. Those design decisions are still subject to further experimentation.

6 Experimental Testing

This chapter describes scope, setup, and results of the demonstrator developed in the course of the project. Based on the requirements described in D2.1 and D2.3 relevant subsets were selected, implemented, and tested as described below.

6.1 Test Setup

The following demonstrator are evaluated:

- Non-privacy-preserving demonstrator
- Privacy-preserving demonstrator
- Privacy-preserving demonstrator with credit handling

See section 6.2 Test Cases for a detailed description of Test Cases that are used to evaluate the various demonstrators.

The following deployment scenarios are considered for the demonstrator evaluation:

- Single instance hosted by Eurocontrol (also called “Centralized Environment”)
- Multiple instances hosted at airports or ANSPs (also called “Hybrid Environment”)
- Decentralized deployment among Airspace Users (also called “Decentralized Environment”)

Finally, concrete data sets are used in the evaluation process of demonstrators and these data sets are made available according to Data Management Plan deliverables (D1.3, D1.6, D1.7):

- Good Cases
- Bad Cases
- Real World Cases provided by SWISS
- NM B2B IF (Network Manager Business-to-Business Interface): implements the FlightListByAerodrome Request/Response flow

While the real-world cases provided by SWISS restrict to a certain regulation Good and Bad Cases

6.2 Test Cases

The functionality of SlotMachine was tested with the following End-to-End tests in the various implementations.

6.2.1 Non-privacy-preserving Demonstrator

The following test steps compromise the non-privacy-preserving demonstrator testing:

Prerequisites:

- these services must be up and running, and version documented for a test run:
  - Controller
  - NMF (service simulating Network Manager)
  - AU REST

- Heuristic Optimizer
- Privacy Engine with MPC Nodes and using unencrypted weights

- AU web app

Tests:

1. Register AU Clients at SlotMachine Instance
   - Login at AU web app
   - Demonstrate connection to Controller

2. Connect SlotMachine to NMF to receive regulations
   - Verify that Controller has NMF instance and regulation specified on start up
   - Verify regular and automatic requests from the Controller to NMF

3. Activate regulation provided by NMF
   - Manually trigger activation of a regulation
   - Verify Controller receives active regulation
   - Verify Controller forwards request to AU

4. Collect inputs from AU
   - Verify AU web client displays active regulation
   - Enter data to participate in an optimization run
   - Submit preferences
   - Verify confirmation from Controller

5. Optimize inputs from AU
   - Verify Controller forwarded initial configuration and AU preferences to Heuristic Optimizer
   - Verify Heuristic Optimizer generates a list of possible flight lists using Privacy Engine (using unencrypted weights)
   - Verify Heuristic Optimizer sends results (list of possible Flight Prioritization Solutions) to Controller

6. Send optimized results to NMF
   - Verify Controller receives list of possible Flight Prioritization Solutions from Heuristic Optimizer
   - Verify Controller forwards list of possible Flight Prioritization Solutions to NMF

7. Distribute selected optimization from NMF to AUs
   - Verify NMF selects and sends Flight Prioritization Solution to Controller
   - Verify Controller informs all effected AUs about Flight Prioritization Solution
   - Verify AU web client displays Flight Prioritization Solution

6.2.2 Privacy-preserving Demonstrator

The following test steps compromise the privacy-preserving Demonstrator testing:

Prerequisites:

- these services must be up and running, and version documented for a test run:
  - Controller
  - NMF (service simulating Network Manager)
  - AU REST (at least 2 instances with one instance simulating an AU; the other instance receives input from the AU web client)
  - PE Encoder for each Airspace User
  - Heuristic Optimizer
  - Privacy Engine and MPC Nodes
  - Dashboard
- Web application for AU Client

Tests:

1. Register AU Clients at SlotMachine Instance
   - Login at AU web client
   - Demonstrate connection to Controller at AU REST instance #1
   - Verify AU REST instance #2 simulating an AU is connected to Controller

2. Connect SlotMachine to NMF to receive regulations
   - Verify that Controller has NMF instance and regulation specified on start up
   - Verify regular and automatic requests from the Controller to NMF

3. Activate regulation provided by NMF
   - Verify NMF has a list of different regulations available
   - Manually trigger activating a specific regulation
   - Verify Controller receives active regulation
   - Verify Controller forwards requests to all affected AUs
   - Verify Dashboard shows active regulation

4. Collect inputs from AUs
   - Verify AU #1 web client displays active regulation
   - Enter data to participate in Slot auction
   - Verify that AU preferences are encryptedSubmit preferences from AU REST #1
   - Verify confirmation from Controller
   - Verify AU REST #2 submits preferences to Controller
   - Verify Dashboard shows number of participating AUs in current regulation

5. Optimize inputs from AUs
   - Verify Controller forwarded initial configuration and both AU preferences to Heuristic Optimizer
   - Verify Heuristic Optimizer only starts optimization after receiving all AU preferences
   - Verify Heuristic Optimizer uses Privacy Engine for processing AU preferences
   - Verify Heuristic Optimizer generates a list of valid Flight Prioritization Sessions
   - Verify Heuristic Optimizer sends results (list of valid Flight Prioritization Sessions) to Controller

6. Send optimized results to NMF
   - Verify Controller receives list of valid Flight Prioritization Solutions

- Verify Dashboard displays number of Flight Prioritization Solutions for active regulation
- Verify Controller forwards list of valid Flight Prioritization Solutions to NMF

7. Distribute selected optimization from NMF to AUs
    - Verify NMF receives valid optimization proposals
    - Verify NMF selects and sends Flight Prioritization Solution to Controller
    - Verify Controller informs all effected AUs about Flight Prioritization Solution
    - Verify Dashboard displays overall gain from Flight Prioritization Solution
    - Verify AU web client displays Flight Prioritization Solution

6.2.3 Privacy-preserving Demonstrator with Credit Handling

The following test steps compromise the privacy-preserving demonstrator with credit handling:

Prerequisites:

- these services must be up and running, and version documented for a test run:
    - Controller
    - NMF (service simulating Network Manager)
    - AU REST (at least three instances with at least two instances simulating AUs; the other instances receive input from AU web clients)
    - PE Encoder for each Airspace User
    - Heuristic Optimizer
    - Privacy Engine and MPC Nodes
    - Blockchain
    - Overview Dashboard
    - AU Dashboard
- Web applications for
    - AU Clients
    - Airports

Tests:

1. Register AU Clients at SlotMachine Instance
    - Login at AU web clients
    - Demonstrate connections to Controller at AU REST
    - Verify AU REST simulating AUs are connected to Controller
    - Verify AU Dashboards show information of respective AU
    - Verify registration is documented on blockchain

2. Connect SlotMachine to NMF to receive regulations
    - Verify that Controller has NMF instance on start up
    - Configure regulation for relevant for test run
    - Verify regular and automatic requests from the Controller to NMF
    - Verify Overview Dashboard display information for SlotMachine instance
    - Verify NMF connection is documented on blockchain

3. Activate regulation provided by NMF
    - Verify NMF has a list of different regulations available

- Manually trigger activating a specific regulation
- Verify Controller receives active regulation
- Verify Controller forwards requests and current optimization session information to all affected AUs
- Verify Overview Dashboard shows active regulation
- Verify regulation status change is documented on blockchain

4. Collect inputs from AUs
    - Verify AU web clients display active regulation
    - Enter data in each AU web client to participate in slot auction
    - Verify that AU preferences are encrypted
    - Submit preferences from AU web clients in encrypted format
    - Verify confirmation from Controller
    - Verify AU REST instances submit preferences to Controller
    - Verify Overview Dashboard shows number of participating AUs in current regulation
    - Verify all AU inputs (partly encrypted) are documented on blockchain

5. Optimize inputs from AUs
    - Verify Controller forwarded initial configuration and all available AU preferences to Heuristic Optimizer at given time
    - Verify Heuristic Optimizer starts optimization according to optimization session information
    - Verify Heuristic Optimizer uses Privacy Engine and MPC nodes for processing AU preferences
    - Verify Privacy Engine validates AU input
    - Verify Heuristic Optimizer generates a list of valid flight lists
    - Verify Heuristic Optimizer sends results (list of valid flight lists) to Controller
    - Verify Heuristic Optimizer operations are documented on blockchain

6. Controller processes flight lists (solutions from optimization run)
    - Verify Controller receives list of valid flight lists
    - Verify Overview Dashboard displays number of Flight Prioritization Solutions for active regulation
    - Verify in credit wallet manager that necessary credits from each AU are available
    - Verify Heuristic Optimizer results and credit verification are documented on blockchain
    - Verify Controller informs all effected AUs and Airports about Flight Prioritization Solutions

7. Check Veto from AUs and Airport
    - Verify AUs and Airports request current Flight Prioritization Solutions
    - Verify AUs and Airports can veto individual items in Flight Prioritization Solutions
    - Verify vetoes are documented on blockchain
    - Verify Controller sends only Flight Prioritization Solutions that are not vetoed to NMF

8. Send optimized results to NMF
    - Verify Controller receives list of valid Flight Prioritization Solutions
    - Verify submitted Flight Prioritization Solutions are documented on blockchain

- Verify Overview Dashboard displays number of Flight Prioritization Solutions for active regulation
- Verify Controller forwards list of valid Flight Prioritization Solutions to NMF

9. Distribute selected optimization from NMF to AUs
    - Verify NMF receives valid optimization proposals
    - Verify NMF selects and sends Flight Prioritization Solution to Controller
    - Verify Controller informs all effected AUs about Flight Prioritization Solution
    - Verify Overview Dashboard displays overall gain from Flight Prioritization Solution
    - Verify AU Dashboard shows updated credit information
    - Verify AU web client displays Flight Prioritization Solution

6.3 Test Runs

In End-to-End tests concrete scenarios are evaluate based on the Test Cases described in the previous section. The following template is used to document Test Runs and results will be available in Deliverable 5.1:

Memo			**Author:** *[N.N.]* **Date:** *[YYYY/DD/MM]*

Subject: Results of Test Run on *[date]*

Configuration & Prerequisites

Test system: *[test system in use (e.g, local deployment, Kubernetes cluster)]*

Services and current images:

- *[name of component: reference to Docker image]*

Datasets:

- *[description of dataset and relevant properties for test run]*

Test cases: *[reference to class of test cases as described in section 7.2]*

Services up and running / auxiliary links:

- *[List of services and configuration including tests how initial functionality was evaluated / smoke test]*
- *[List of UI clients including credentials]*

[X. Enumerated List of Test Cases (number of successful tests/overall test number)]

- *[Test Step – Test Result (OK / NOK)]*

Observations

Test Run Summary: *[number of successful tests/overall test number]*

[each component]

- *[findings (bugs, comments)]*

-- End of Memo –

7 Conclusions

This document provided a first comprehensive design of the SlotMachine system. The design is based on the concept of a microservice architecture in order to allow for horizontal scaling and distributed processing in the cloud; where possible, open technologies shall be used to build the platform. In particular, this document describes the main components of the SlotMachine system as well as the interfaces of these components along with the interaction between the components in form of UML sequence diagrams. Some of the design decisions are deferred to later stages in the project, in particular those related to credit handling as well as the communication between Privacy Engine and MPC nodes. The former is dependent on the market mechanisms and also requires further experimentation. Likewise, the latter is still subject to further experimentation. Nevertheless, this document provides a good basis for further implementation and experimentation in the course of the SlotMachine project.

8 References

[1] SESAR, “D2.1 - Requirements Specification,” SlotMachine, 890456, 2021.

[2] SESAR, “D2.3 - Business Concepts,” SlotMachine, 890456, 2021.

[3] SESAR, “D3.2 - Specification of the PrivacyEngine Component,” SlotMachine, 890456, 2021.

[4] SESAR, “D4.2 - Specification of Evolutionary Algorithm,” SlotMachine, 890456, 2021.

[5] SESAR, “D5.1 - SlotMachine Platform Demonstrator,” SlotMachine, 890456, 2021.

[6] S. Allamaraju, RESTful web services cookbook: solutions for improving scalability and simplicity, O'Reilly, 2010.

[7] EUROCONTROL, NM 25.0 - NM B2B Reference Manuals - Flight-Services, Brussels: EUROCONTROL, 2021.

[8] SESAR, “D4.1 - Report on State of the Art of Relevant Concepts,” SlotMachine, 890456, 2021.

[9] SESAR, “D3.1 - Report on State of the Art of Relevant Concepts,” SlotMachine, 890456, 2021.

Appendix A Terms of Glossary

Term	Definition
Airport slot	Permission to operate a service at a given time (with regard to flight arrivals and departures).
Airspace User (AU)	Airlines and low volume users (LVUC) including Business aviation.
Air traffic flow management (ATFM) slot	ATFM departure slot, assigned tactically by the Network Manager to manage congestion.[1]
Baseline Delay	Amount of delay a flight or group of flights would be assigned if no UDPP prioritisation were applied.
Cancellation	A flight is cancelled if it is not operated (usually reallocating the passengers to other flights). See also 'suspension'.
Capacity	Maximum number of flights that can enter into a sector or airport per unit of time (usually 1 hour, but can be defined for any time-window length, e.g., 15 minutes).
Capacity Constrained Situation	A period of capacity and demand imbalance in which the capacity reduction and the resulting excess demand causes stress in the ATFM slot allocation process, relative to that allocated previous to the imbalance.
Capacity surplus	Difference between capacity and demand when the available capacity is enough to allocate the actual demand for a given period of time and there is still room to potentially allocate a higher number of flights.
Cost of delay	Economic cost incurred by an AU due to the delay experienced by a flight.
Credit	For ease of reading, in this document, the term 'credit' will often be used as a substitute for 'Delay Credit'.
Delay	The difference between the ATFM slot and the scheduled time of departure.

[1] https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=LEGISSUM:tr0032&from=EN

Term	Definition
Delay Credit	Unit of the virtual currency (i.e. credits) as an expression of the value generated to other users (positive externalities) or the loss caused (negative externalities) in terms of delay.
Demand	Total number of flights that plan to enter into a sector or airport in a given unit of time (usually defined for 1 hour but can be also defined for any time-window length, e.g. 15 minutes).
Equity	Equity, measures how uniformly the distribution of the good is performed, that is, without taking into account individual satisfaction thresholds.
Excess demand	Difference between demand and capacity when the available capacity is not enough to allocate the demand for a given period of time.
Externalities	Collateral effects caused by the decision/action of an AU regarding the usage (or non-usage) of a certain slot. When such decisions change (limiting or expanding), it allows other AU to use the same or other slots.
Fairness	Fairness can be defined as the quality of distributing something among a set of individuals in a manner such that each receives a share that fulfils its individual satisfaction threshold. In order to measure fairness objectively, it is essential to agree on a common way to quantify such individual satisfaction thresholds.
Flight list	A sequence of flights.
Flight prioritization session	A group of departing aircrafts in a certain time span that is optimized for their starting sequence based on AU inputs. Also often referred to as optimization session.
Flight prioritization (solution)	Best possible solution found by SlotMachine within a flight prioritization exchange session.
Hotspot	Similar to CCS, but while CSS refers to periods of demand and capacity imbalance at airports, HSPT refers to periods of demand and capacity imbalance in en-route sectors.

Term	Definition
Multiparty Computation (MPC)	Secure multiparty computation (MPC / SMPC) is a cryptographic protocol that distributes a computation across multiple parties where no individual party can see the other parties' data.[2]
Near on-time	Delay higher than 0' but ≤ 15' for an AU.
Network Manager (NM)	Network Manager - the function is currently attributed to ECTL. The B2B services are provided by the NM.
Network Management Functions (NMF)	This is the set of functions for managing the safety of ATM network operations that are either delegated to other ATM actors such as local DCB (Demand-Capacity Balancing) by Flow Manager (FMP) or at Airport; AUs; or performed by the Network Manager itself.
Network Manager	The body entrusted with the tasks necessary for the execution of the functions referred to in Article 6 of Regulation (EC) No 551/2004.[3]
On-time	Delay = 0' for an AU.
Operational Service and Environment Description (OSED)	The Operational Service and Environment Definition (OSED) document describes the operational concept defined in the Detailed Operational Description (DOD) in the scope of its Operational Focus Area (OFA). It defines the operational services, their environment, scenarios and use cases and requirements.[4]
Protection	A flight is protected if the AU sends a request (manually or through automated means) to allocate the flight to a particular slot, usually between the scheduled time (i.e., zero delay) and the slot assigned by FPFS (i.e., the Baseline Delay). The flight is *actually* protected only if no other flights with higher priority express the desire of using the same slot (i.e., protect their flights in the same slot).

[2] https://en.wikipedia.org/wiki/Secure_multi-party_computation

[3] https://www.skybrary.aero/index.php/Network_Manager

[4] https://www.sesarju.eu/sites/default/files/documents/solution/Sol107%204%20Point%20Merge%20Complex%20TMA_OSED.pdf


SlotMachine


SESAR
JOINT UNDERTAKING


AIT
AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY


FREQUENTIS
FOR A SAFER WORLD


JKU
JOHANNES KEPLER
UNIVERSITÄT LINZ


SWISS