2024

UPCAST PROJECT

Draft Document

DELIVERABLE 2.2

THIS DOCUMENT IS IN DRAFT FORM AND PENDING OFFICIAL APPROVAL. IT IS SUBJECT TO REVIEW AND MAY BE UPDATED.

D2.4: Privacy and Usage Control Modules

This project has received funding from the European Union's Horizon Research and Innovation Actions under Grant Agreement № 101093216.

Title:	Document version:
D2.4 Privacy and Usage Control Modules I	0.1

Project number:	Project Acronym	Project Tittle
101093216	UPCAST Project	UPCAST Project

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type*-Security*:
M15 (March 2024)	M15 (March 2024)	OTHER - PU
Responsible:	**Organization:**	**Contributing WP:**
Luis-Daniel Ibáñez	University of Southampton	WP2

Authors (organization):

Luis-Daniel Ibáñez (Soton)

Jaime Salas (Soton)

Tek Raj Chhetri (Soton)

Semih Yumusak (Soton)

George Konstantinidis (Soton)

Mariza Koukovini (Abovo)

Eugenia Papagiannakopoulou (Abovo)

George Lioudakis (Abovo)

Majid Ektesabi (Nokia)

Abstract:

The UPCAST project is a pioneering endeavour that enhances data sharing across diverse sectors providing Universal Plugins to data marketplaces that provide or improve fundamental building blocks of data sharing platforms. This deliverable describes the first version of the Privacy and Usage

Constraint and the Federated Machine Learning plugins

Keywords:

Privacy, Usage Constraints, Data Operations, Data Processing Workflows,

REVISION HISTORY

Revision:	Date:	Description:	Author (Organization)
V0.1	15.10.2024	Initial version	Majid Ektesabi(Nokia)

This project has received funding from the European Union's Horizon Research and Innovation Actions under Grant Agreement Nº 101093216.

More information available at https://upcastproject.eu/

COPYRIGHT STATEMENT

INDEX

LIST OF FIGURES

1 Purpose and scope

This deliverable presents the first version of the Privacy and Usage Constraints, and Federated Machine Learning plugins, linked to tasks T2.1 and T2.2 in Work Package 2. We also present relevant advances on the Data Processing Workflow plugin (DPW), such as specification and evaluation of usage constraints against DPW purposes.

It serves as a public record of the progress achieved in these tasks, showcasing the plugin development through practical demonstrations.

Deliverable D1.2 in Work Package 1 defined the Upcast MVP and its core functionalities. Figure 1 below highlights the plugins in the scope of this deliverable.

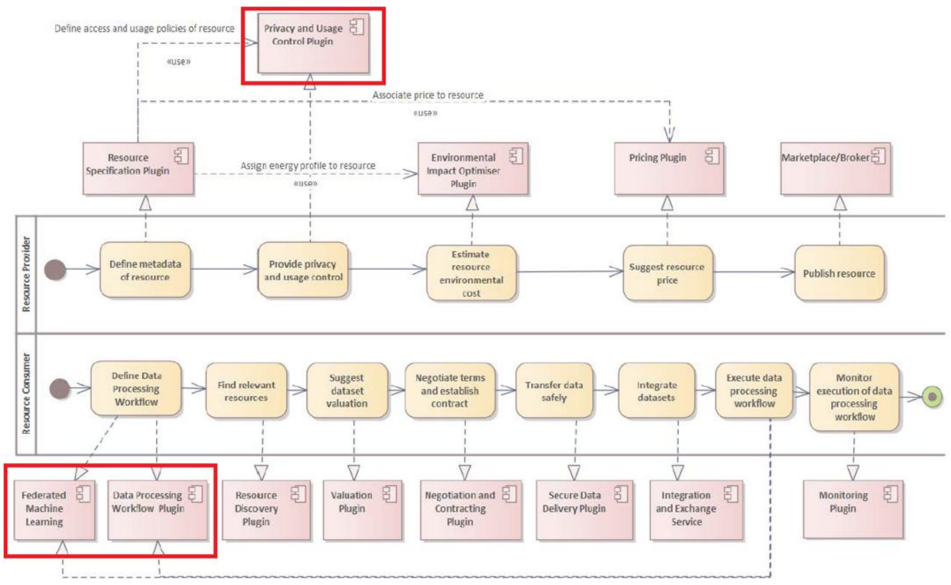

Define access and usage policies of resource
Privacy and Usage Control Plugin
«use»
Associate price to resource
«use»
Resource Specification Plugin
Assign energy profile to resource
«use»
Environmental Impact Optimiser Plugin
Pricing Plugin
Marketplace/Broker
Resource Provider
Define metadata of resource
Provide privacy and usage control
Estimate resource environmental cost
Suggest resource price
Publish resource
Resource Consumer
Define Data Processing Workflow
Find relevant resources
Suggest dataset valuation
Negotiate terms and establish contract
Transfer data safely
Integrate datasets
Execute data processing workflow
Monitor execution of data processing workflow
Federated Machine Learning
Data Processing Workflow Plugin
Resource Discovery Plugin
Valuation Plugin
Negotiation and Contracting Plugin
Secure Data Delivery Plugin
Integration and Exchange Service
Monitoring Plugin

Figure 1: UPCAST MVP overview. Plugins in the scope of this deliverable: Privacy, Data Processing Workflow and Federated Machine Learning Plugins highlighted in red.

Section 2 focuses on the Privacy and Usage Constraint Plugin. The presentation of the plugin is divided in three sub-components: Definition of privacy and usage constraints by a Data Producer (Section 2.1.1); Soliciting consent and user specific constraints (Section 2.1.2); and Data Consumer internal policies and Data Processing Workflows (Section 2.1.3). The latter includes the advances on Data Processing Workflows specification and assessment.

Section 3 focuses on the Federated Machine Learning Plugin. Section 4 details conclusions and next steps.

2 Privacy and Usage Constraint Plugin

This plugin supports both Data Producers and Data Consumers in specifying fine-grained usage constraints.

Based on the specifications delivered in D1.1 and D1.2, we further refined two general usage scenarios of Privacy and Usage Constraint featuring a Data Producer and a Data Consumer:

Scenario 1:

1. A Data Producer defines privacy and usage constraint rules on a dataset they wish to publish or advertise on a Data Marketplace. In this scenario we assume the dataset does not include any personally identifiable information, hence, the Data Producer is the sole owner.
2. A Data Consumer specifies a Data Processing Workflow where one stage may make use of a Dataset like the one specified in step 1. The Data Consumer goes to a Data Marketplace and discovers[1] the Dataset specified in step 1. The privacy and usage constraints defined by the Data Producer in step 1 are checked against the steps in the DPW and any conflicts are detected and informed. If there are no conflicts, the Consumer may choose to acquire the dataset. If there are conflicts, the Consumer may either discard the Dataset or request a negotiation with the Data Producer through the Negotiation plugin. The Negotiation plugin will be detailed in Deliverable 3.1 as part of WP3.
3. For cases where a Data Consumer requires personal data from individuals that do not publish in a Marketplace, this plugin provides a solution for the Data Consumer to generate a form to request informed consent and record responses from the individuals. Consumers are also able to specify use case specific constraints additional to the consent essential to the intended data processing, that Subjects can grant or revoke. For example, "permission to re-sell data to third parties" is not essential to data processing, but desirable to ask individuals.

[1] In this deliverable, we abstract from details of how discovery happens. Refer to D2.1 for details on the UPCAST Discovery plugin.

Scenario 2:

1. This scenario starts from step 3 of scenario 1: a Data Consumer requiring personal data from individuals that do not have it advertised in a Marketplaces, generate forms to request informed consent, and record responses. Consumers are also able to specify use case specific constraints additional to the consent essential to the proceeding, that Subjects can grant or revoke (e.g. "permission to further re-sell data to third parties")
2. A Data Consumer may assume the role of a Data Producer and publish/advertise on a Data Marketplace a dataset comprised of an appropriate subset of the data collected in step 1, e.g., those from individuals that consented to further re-sell data to third parties. Additionally, they may want to define additional usage constraint rules such as commercial usage restrictions or geographical place of processing.
3. As per Step 2 of scenario 1, with the addition that conflict resolution is also executed against individual use case specific constraints, e.g., if an individual allowed re-sale of their data but only for "Research" purposes, and the Processing Workflow's purpose is for "Marketing", a conflict is flagged.

We demonstrate Scenario 2. Steps 1 and 2 are demonstrated on recording "Consent Forms and Data Usage Constraints", Step 3 is demonstrated on recording "Data Consumer perspective". In this deliverable, we also illustrate the general scenarios with a running example stemming from the Health and Fitness use case described in section 4.3 of D1.1. In this use case consortium partner NISSA acts as a Data Consumer that collects personal data from individuals (trainees) that engage in fitness activities using smart wearables. Data collected includes heart rate and acceleration vectors with the purpose of gaining insights into the trainees' physiological response, intensity of the exercise, recovery patterns and overall cardiovascular fitness. This data can be used to provide personalised reports to the trainees (e.g. statistical analysis or estimation of number of calories burnt) or engage in social activities around training such as challenges and leaderboards. NISSA may also create aggregated analysis from trainees data and NISSA may also create aggregations of trainees data further augmented with data from sensors installed in exercise machines and create a dataset they could advertise on a marketplace, becoming a Data Producer.

Figure 2 shows the use cases of this plugin specified in D1.2.

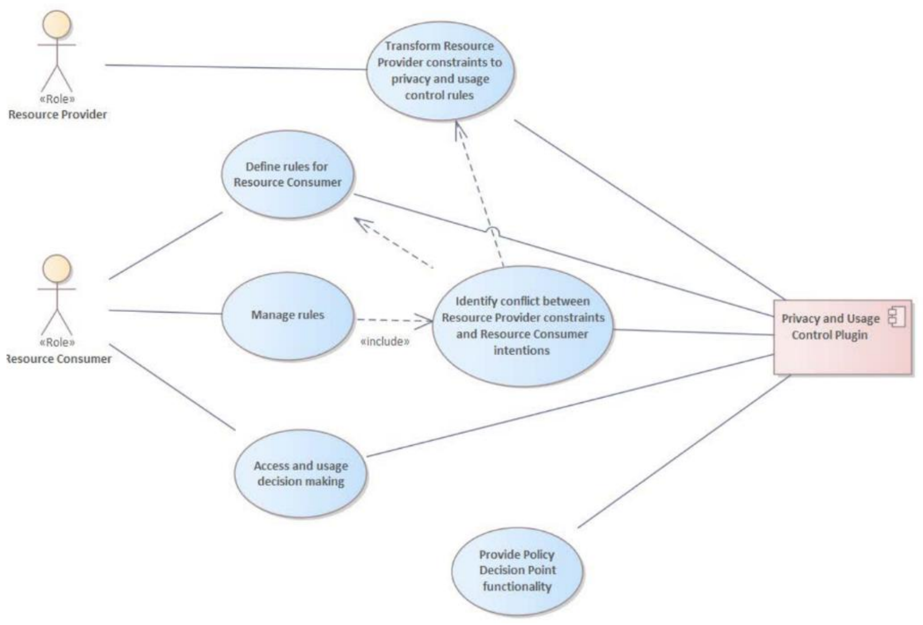

«Role»
Resource Provider
«Role»
Resource Consumer
Transform Resource Provider constraints to privacy and usage control rules
Define rules for Resource Consumer
Manage rules
«include»
Identify conflict between Resource Provider constraints and Resource Consumer intentions
Access and usage decision making
Provide Policy Decision Point functionality
Privacy and Usage Control Plugin

Figure 2: Use case diagram of Privacy and Usage Control plugin as per D1.2

At the time of this deliverable, we have achieved full implementation of use cases "Define Rules for Resource Consumer", "Manage Rules" and "Provide Policy Decision Point Functionality", and partial implementation of "Identify Conflicts" and "Transform Resource Provider Constraints", that will be completed for D2.4.

2.1 Steps to run the demonstrator

2.1.1 Data Producer defines privacy and usage constraint rules

In this section of the demonstration we explain how a Data Producer defines privacy and usage constraint rules on their dataset. The main interface is depicted in Figure 3, it's a form that contains several dropdown menus and a text input where we define the terms that comprise a rule:

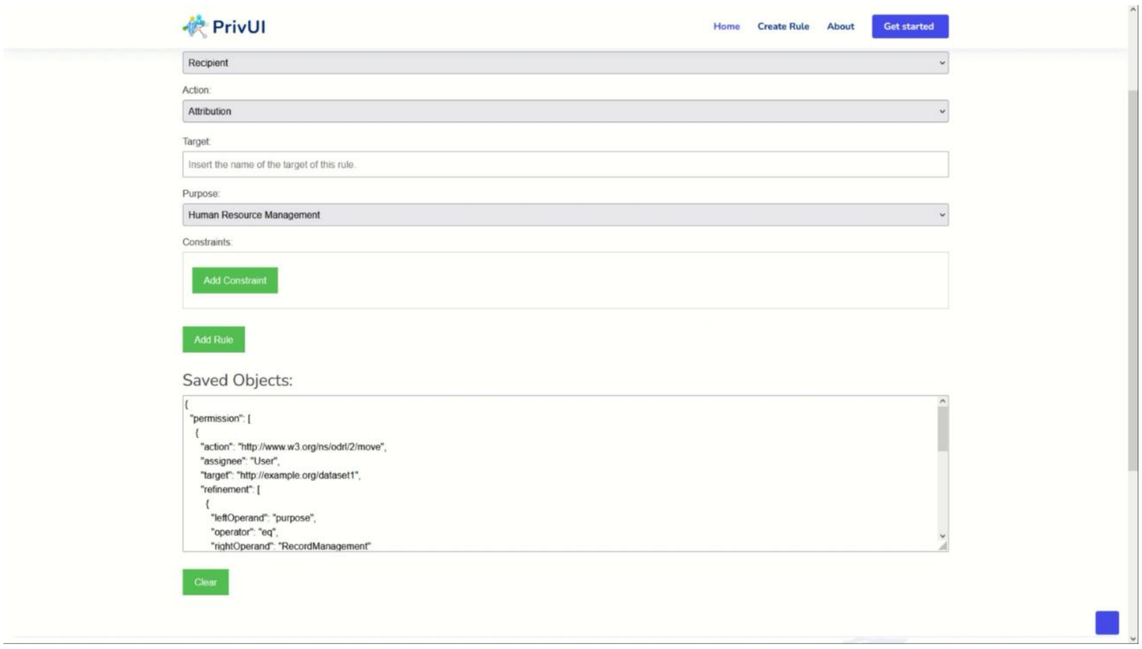
- The type of rule, whether it be a permission, a prohibition or an obligation.
- The actor or entity who this rule applies to. These entities come from DPV and denote entities that have been defined in GDPR.
- The target of this rule, which is an asset that is the subject of this rule.
- The action or data operation that is applied to the target of this rule.
- The purpose of this rule, which is a general purpose of a data processing task as defined in DPV.

In addition, we have identified a specific use-case, where the user constructs a prohibition on the action “Execute”. Selecting these two values will make a text area pop-up under the action dropdown menu, where the user can specify a query in some standard query language (such as SQL). This states that the actor cannot execute the specified query over the target of the rule, while following the other terms of the rule normally.

Furthermore, the form contains a field where any number of additional constraints can be added (by clicking on the “Add Row” button) or removed (by clicking on the “Remove” button next to each constraint) to make more expressive rules. Each constraint is comprised by three values: a left operand that denotes the semantics of this constraint (e.g., datetime means this constraint is about a specific date and time), an operator that compares values or checks for inclusions, and a right operand that denotes a specific value. If a rule contains multiple constraints, each constraint must be fulfilled for the rule to be valid.

Once all values have been selected, pressing the “Add Rule” button will output an ODRL policy describing the constructed rule in a text area. Additionally, this will reset the form to allow the user to define more rules in the same manner. Pressing the “Add Rule” button when the text area already contains rules will append the current rule to the saved policy and output an ODRL policy describing the new rule as well as the previous ones. Finally, pressing the “Clear” button will clear the text area of any rules, allowing the user to start over.

In the Health and Fitness example, partner NISSA is interested in limiting the potential use of the dataset they created mashing up smart wearables and smart exercise machine data for the purpose of research and development of exercise machines.



Recipient

Action:

Attribution

Target:

Insert the name of the target of this rule.

Purpose:

Human Resource Management

Constraints:

Saved Objects:

```
{
 "permission": [
  {
   "action": "http://www.w3.org/ns/odrl/2/move",
   "assignee": "User",
   "target": "http://example.org/dataset1",
   "refinement": [
    {
     "leftOperand": "purpose",
     "operator": "eq",
     "rightOperand": "RecordManagement"
```

Figure 3 Interface for usage constraint rule Creation. Data Producers may declare constraints for "Actions" on "Target" data for a given "Purpose". The lower panel shows the rule in Machine-Readable format.

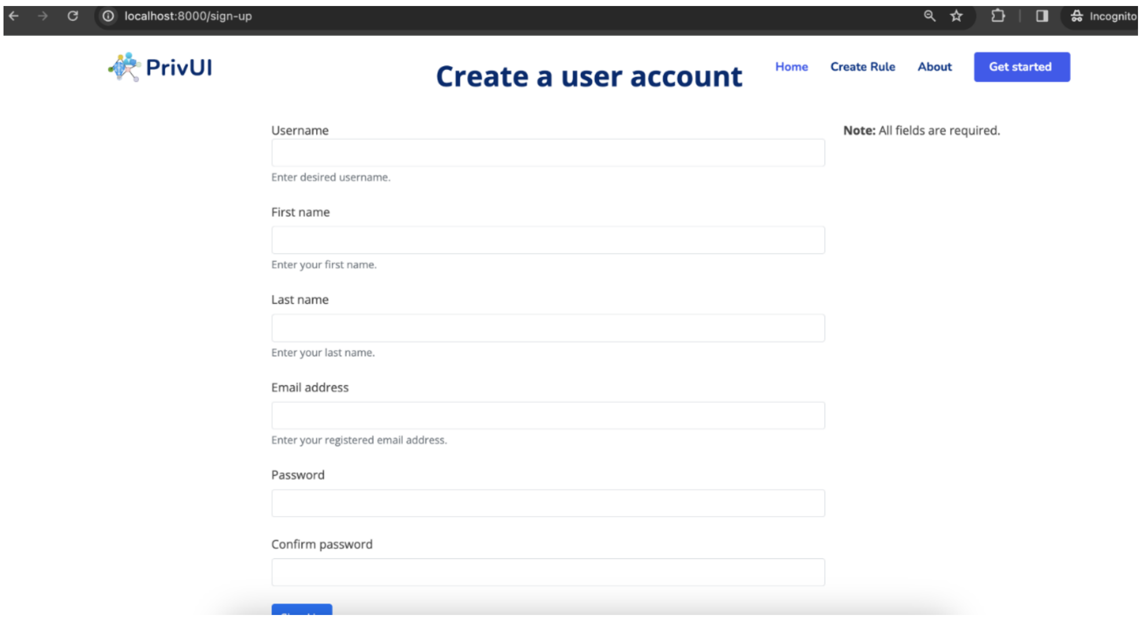
2.1.2 Soliciting Consent and Use Specific Constraints

UPCAST's PrivUI (see Figure 1) is designed to be used by both the data consumer and data producer. The data consumer can also be referred to as the data controller/processor and data producer as data subject in GDPR terminology. Section 2.3.1 describes the functionalities provided by the PrivUI for the data consumer and Section 2.3.2.2 for the data provider.

Data consumer view

The consumer can configure the consent form along with the additional use case specific constraints to request the consent from the data producer. In order to configure the consent form, the following steps must be followed.

1. The first step is to register as a user of type 'data consumer' in the PrivUI (Figure 4). By default, the user is registered as data producer, which needs to be validated and changed to the type 'data consumer' by the admin user.



Create a user account

Note: All fields are required.

Username

Enter desired username.

First name

Enter your first name.

Last name

Enter your last name.

Email address

Enter your registered email address.

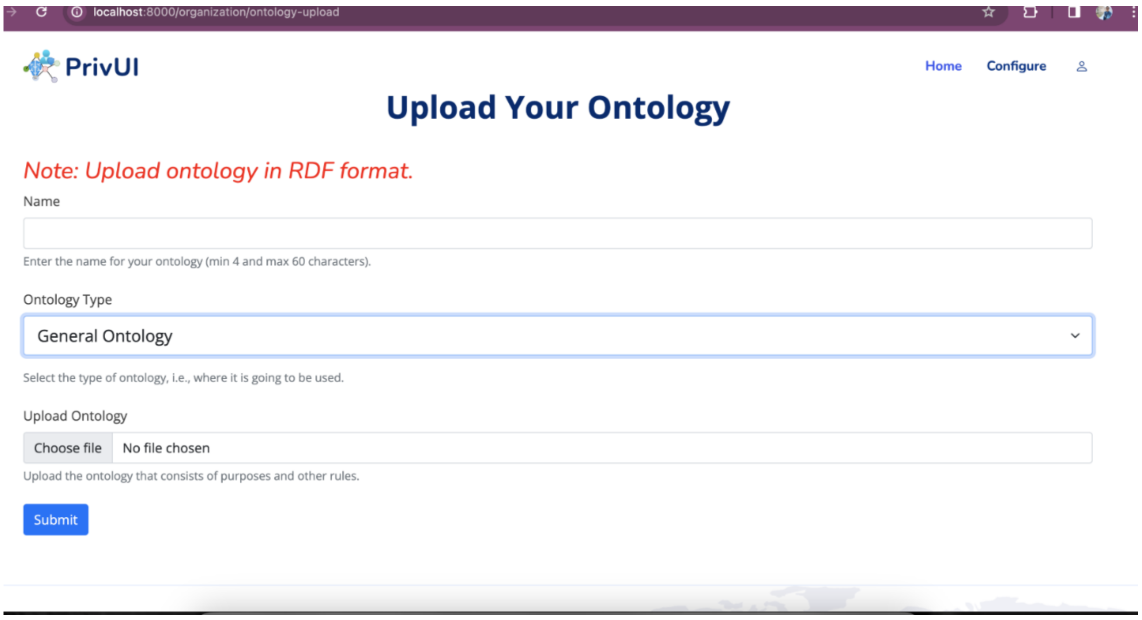
Password

Confirm password

Figure 4: PrivUI user registration

2. Next login to the portal, i.e., PrivUI and navigate to ***Configure→Upload Ontology*** menu option, where you can upload three different types of ontology, namely, (i) General Ontology; (ii) Domain Ontology; and (iii) Data Schema Ontology. By default, DPV and ODRL ontologies are used as general ontology and therefore unless you want you have your own general ontology there's no need to upload it. However, the

other two ontologies, Domain and Data Schema ontologies needs to be uploaded. In our running example with the Health and Fitness usecase, the Domain ontology describes terms and taxonomies related to health and fitness, for example the specific purpose of “Development of Exercise Machines”, while the Data Schema corresponds to the collected data: heart rate and acceleration vectors. Figure 5 shows the Upload ontology page of PrivUI. Additionally, you can also view the uploaded ontology as shown in Figure 6 by navigating to the menu, **Configure→View Ontology**.



Upload Your Ontology

Note: Upload ontology in RDF format.

Name

Enter the name for your ontology (min 4 and max 60 characters).

Ontology Type

General Ontology

Select the type of ontology, i.e., where it is going to be used.

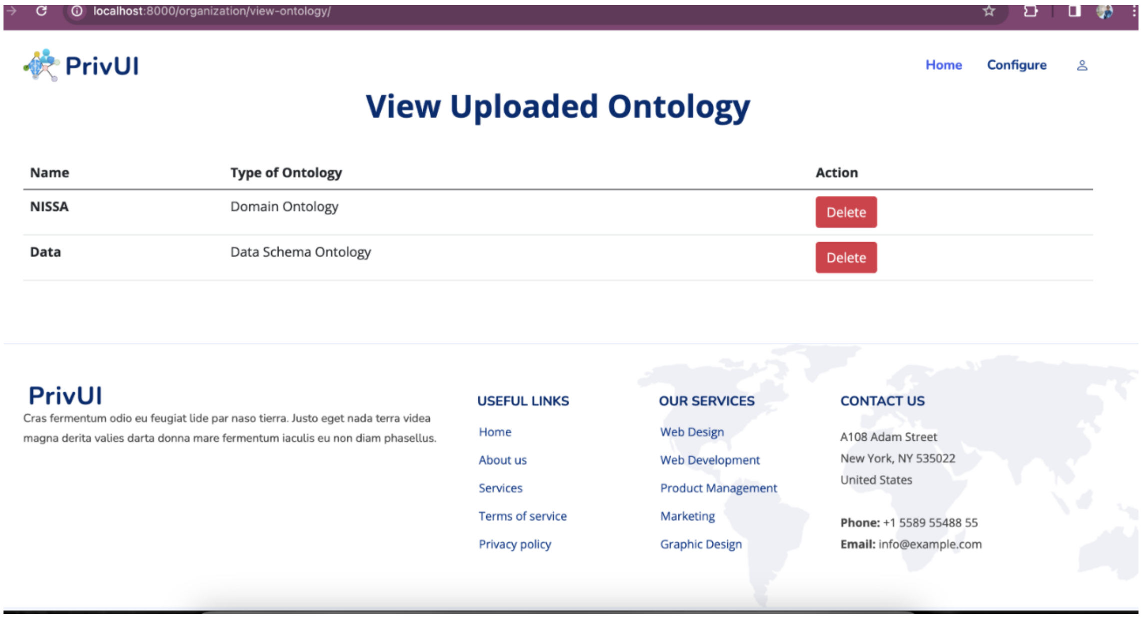
Upload Ontology

Choose file No file chosen

Upload the ontology that consists of purposes and other rules.

Submit

Figure 5: PrivUI Ontology Upload interface



View Uploaded Ontology

Name	Type of Ontology	Action
NISSA	Domain Ontology	Delete
Data	Data Schema Ontology	Delete

PrivUI

Cras fermentum odio eu feugiat lide par naso tierra. Justo eget nada terra videa magna derita valies darta donna mare fermentum iaculis eu non diam phasellus.

USEFUL LINKS
Home
About us
Services
Terms of service
Privacy policy

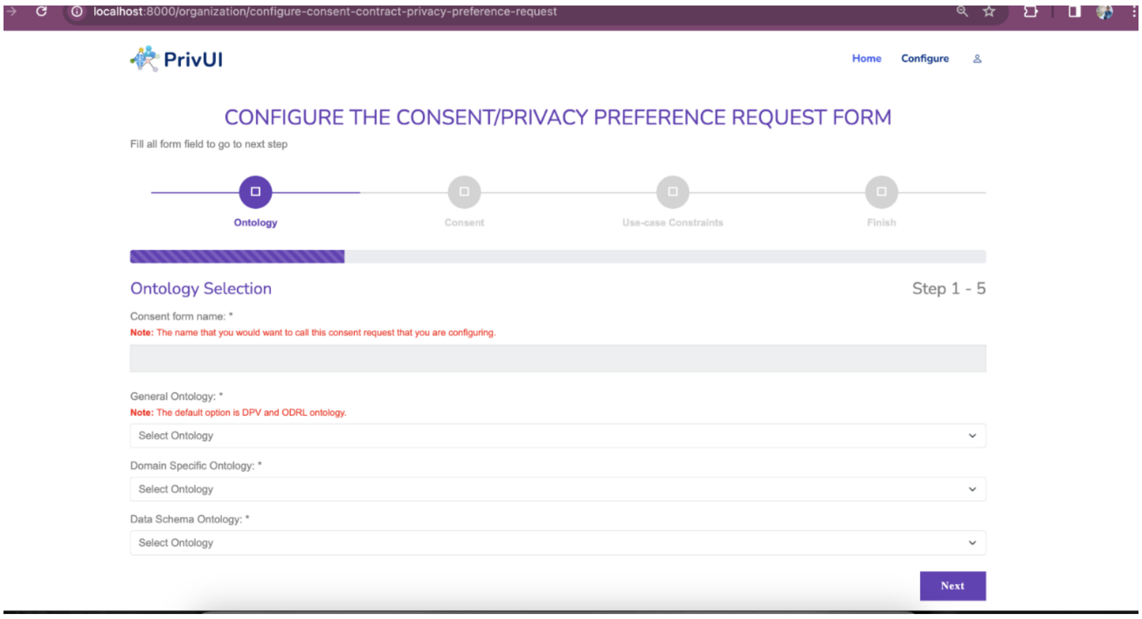
OUR SERVICES
Web Design
Web Development
Product Management
Marketing
Graphic Design

CONTACT US
A108 Adam Street
New York, NY 535022
United States

Phone: +1 5589 55488 55
Email: info@example.com

Figure 6: PrivUI View Ontology page. The NISSA in the figure is the provided name of the uploaded domain ontology, representing Nissatech use case. Similarly, is the case for the data schema ontology

3. After successfully uploading all the required ontologies, navigate to the menu **Configure→Configure Data Request** to configure the consent request. This page consists of a form with step-by-step instructions that need to be filled out with the information, such as the purpose for which data is being collected, the duration, and the data being collected, all of which are required for collecting and processing personally identifiable (PII) data as per GDPR. Additionally, the page also allows to specify use case specific constraints. Figures 7, 8 and 9 shows the steps involved in the configuration of the consent request.
4. Once successfully configured, the form can be sent to the data providers by selecting the list of the users (or data providers) in order to collect the data provider consent response as shown in Figure 10. The page can also be navigated via menu **Configure→Send Consent Request.** In our Health and Fitness use case running example, trainees receive the form and can grant/revoke consent about the use of their heart rate and acceleration vector data for the purposes stated by the Data Consumer NISSA.
5. The user, i.e., data provider response to the consent request can be viewed in ODRL format by navigating to the menu **Configure → View ODRL Response.**


localhost:8000/organization/configure-consent-contract-privacy-preference-request

CONFIGURE THE CONSENT/PRIVACY PREFERENCE REQUEST FORM

Fill all form field to go to next step

Ontology Selection Step 1 - 5

Consent form name: *

Note: The name that you would want to call this consent request that you are configuring.

General Ontology: *

Note: The default option is DPV and ODRL ontology.

Select Ontology

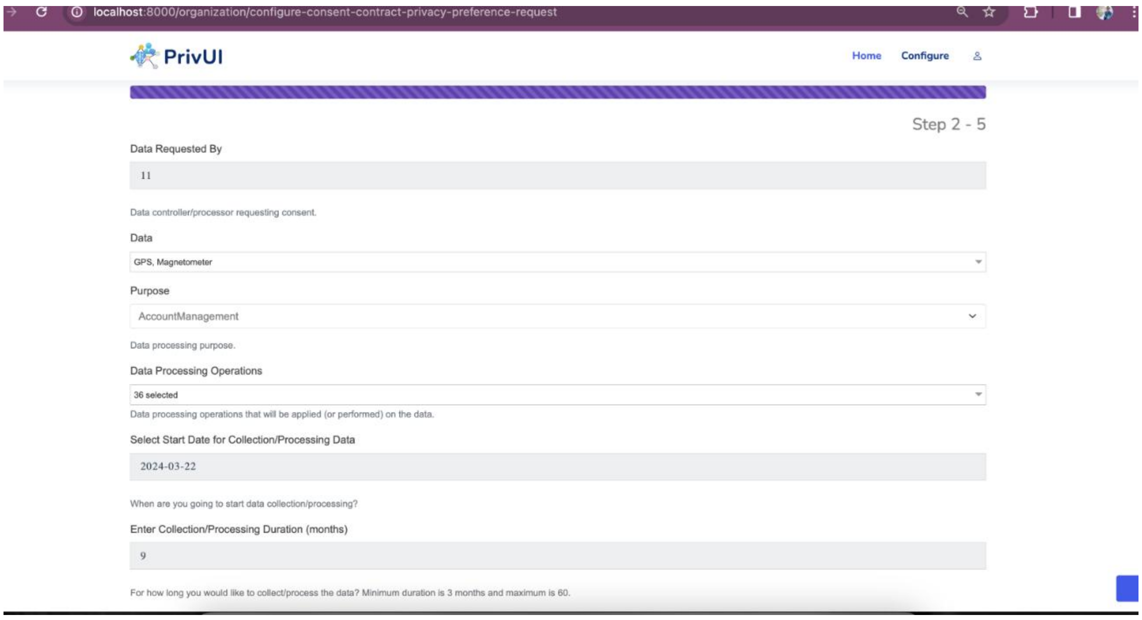
Domain Specific Ontology: *

Select Ontology

Data Schema Ontology: *

Select Ontology

Figure 7: Ontology selection for consent form configuration


localhost:8000/organization/configure-consent-contract-privacy-preference-request

Step 2 - 5

Data Requested By

11

Data controller/processor requesting consent.

Data

GPS, Magnetometer

Purpose

AccountManagement

Data processing purpose.

Data Processing Operations

36 selected

Data processing operations that will be applied (or performed) on the data.

Select Start Date for Collection/Processing Data

2024-03-22

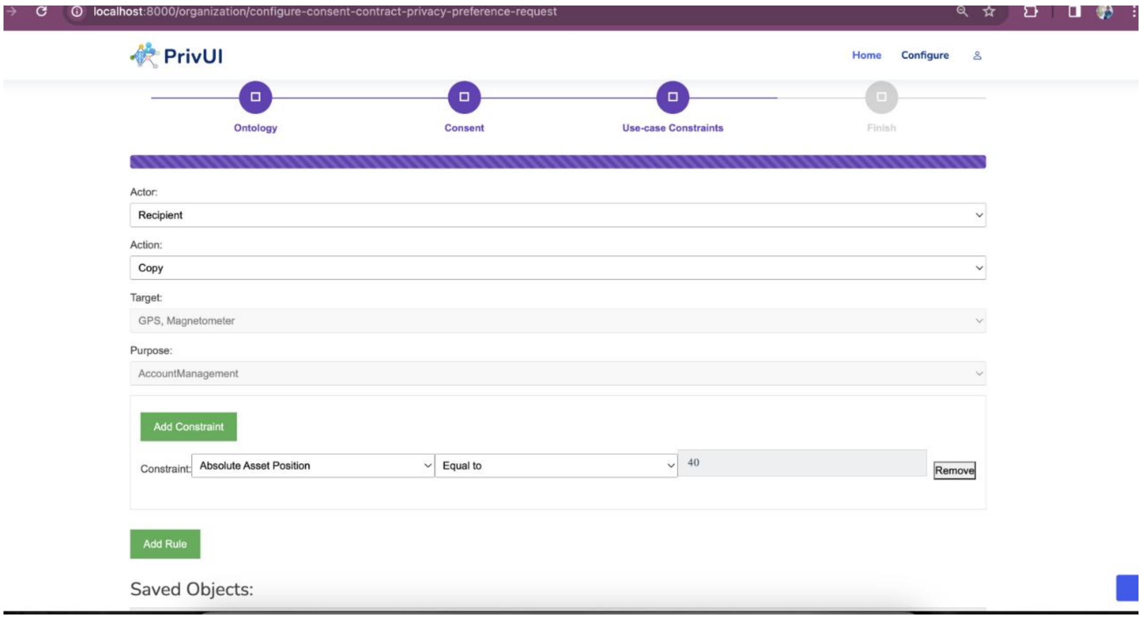
When are you going to start data collection/processing?

Enter Collection/Processing Duration (months)

9

For how long you would like to collect/process the data? Minimum duration is 3 months and maximum is 60.

Figure 8: Details of consent form configuration


localhost:8000/organization/configure-consent-contract-privacy-preference-request

PrivUI — Home | Configure

Ontology — Consent — Use-case Constraints — Finish

Actor: Recipient

Action: Copy

Target: GPS, Magnetometer

Purpose: AccountManagement

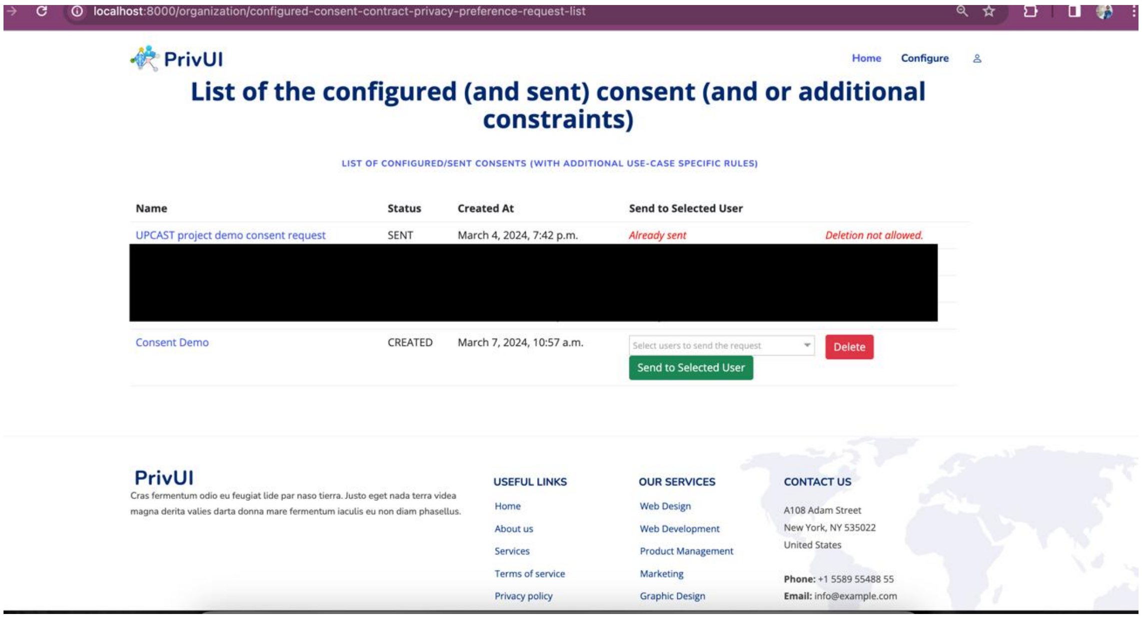
Add Constraint

Constraint: Absolute Asset Position | Equal to | 40 | Remove

Add Rule

Saved Objects:

Figure 9: Additional use case specific constraints configuration


localhost:8000/organization/configured-consent-contract-privacy-preference-request-list

PrivUI — Home | Configure

List of the configured (and sent) consent (and or additional constraints)

LIST OF CONFIGURED/SENT CONSENTS (WITH ADDITIONAL USE-CASE SPECIFIC RULES)

Name	Status	Created At	Send to Selected User	
UPCAST project demo consent request	SENT	March 4, 2024, 7:42 p.m.	Already sent	Deletion not allowed.
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Consent Demo	CREATED	March 7, 2024, 10:57 a.m.	Select users to send the request / Send to Selected User	Delete

PrivUI
Cras fermentum odio eu feugiat lide par naso tierra. Justo eget nada terra videa magna derita valies darta donna mare fermentum iaculis eu non diam phasellus.

USEFUL LINKS: Home, About us, Services, Terms of service, Privacy policy

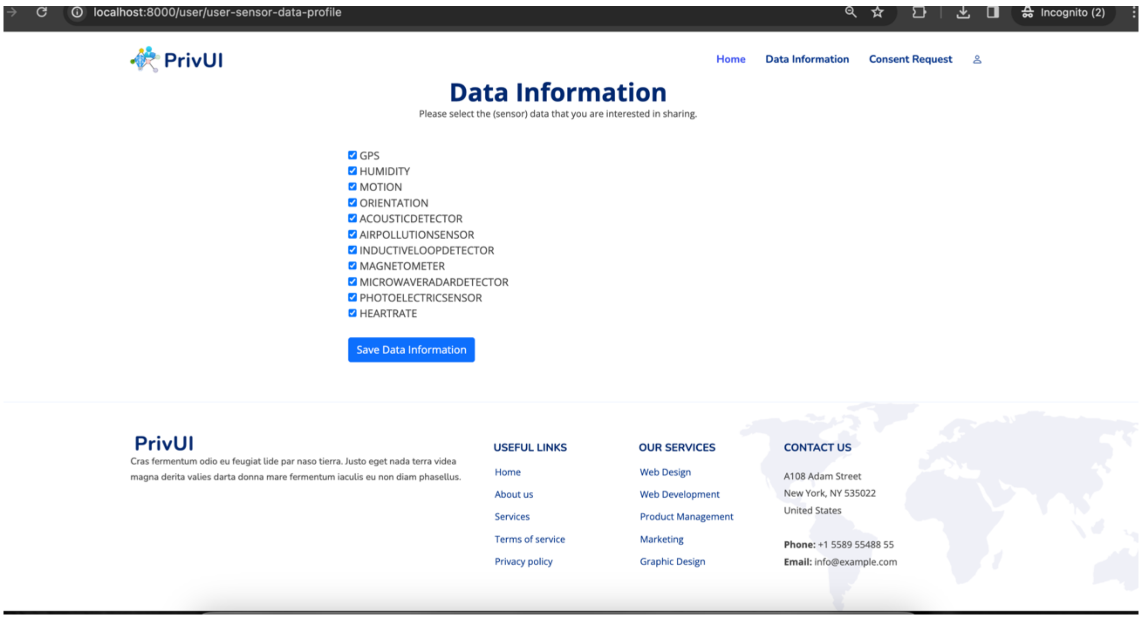
OUR SERVICES: Web Design, Web Development, Product Management, Marketing, Graphic Design

CONTACT US: A108 Adam Street, New York, NY 535022, United States
Phone: +1 5589 55488 55
Email: info@example.com

Figure 10: Submitting consent request to data provider (individual)

Data producer view

UPCAST's PrivUI provides the data producer or data subject the functionalities to configure the data that they are interested in sharing and provide permission (or prohibition) to the requested consent. Figure 11 shows the data information that the user is interested in sharing with the data consumer for all sensors being collected in the Health and Fitness use case. This data information will be matched against the data information from the consent request form prior to sending the request to the selected user (or data provider). Similarly, Figure 12 shows the list of the consent requests received by the users and their status, i.e., whether the consent was given or revoked. In a similar manner, Figure 13 provides a view of the selected consent request along with use case-specific custom constraints and an option to provide consent (or permission) through an opt-in or opt-out method via a checkbox.



localhost:8000/user/user-sensor-data-profile

Data Information

Please select the (sensor) data that you are interested in sharing.

- ✓ GPS
- ✓ HUMIDITY
- ✓ MOTION
- ✓ ORIENTATION
- ✓ ACOUSTICDETECTOR
- ✓ AIRPOLLUTIONSENSOR
- ✓ INDUCTIVELOOPDETECTOR
- ✓ MAGNETOMETER
- ✓ MICROWAVERADARDETECTOR
- ✓ PHOTOELECTRICSENSOR
- ✓ HEARTRATE

PrivUI

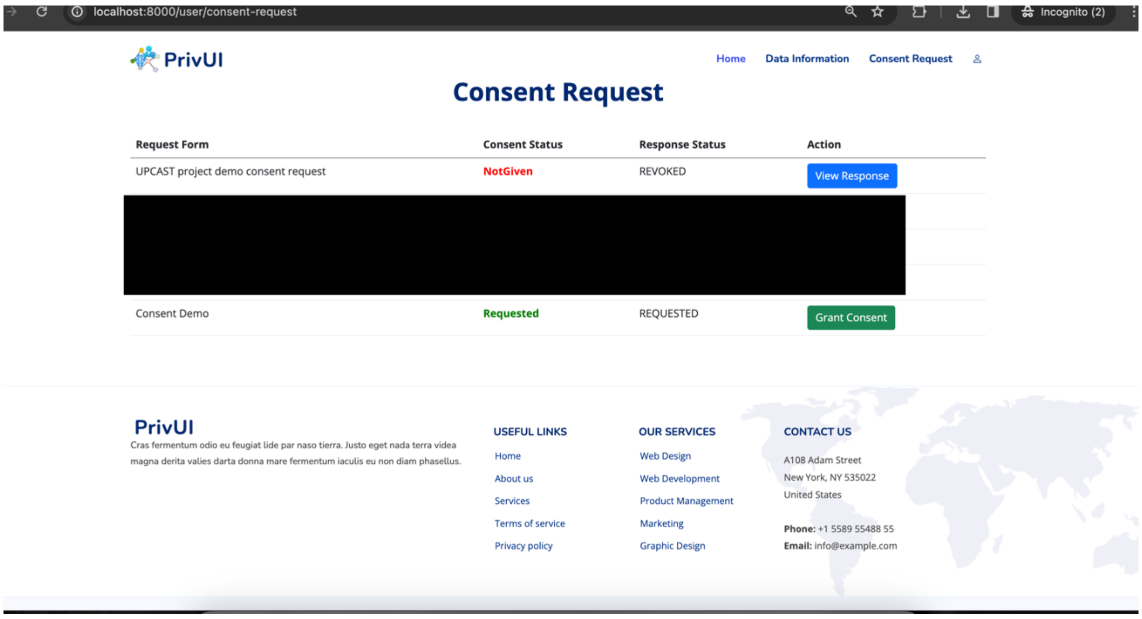
Cras fermentum odio eu feugiat lide par naso tierra. Justo eget nada terra videa magna derita valies darta donna mare fermentum iaculis eu non diam phasellus.

CONTACT US

A108 Adam Street
New York, NY 535022
United States

Phone: +1 5589 55488 55
Email: info@example.com

Figure 11: Data producer configuration of data information



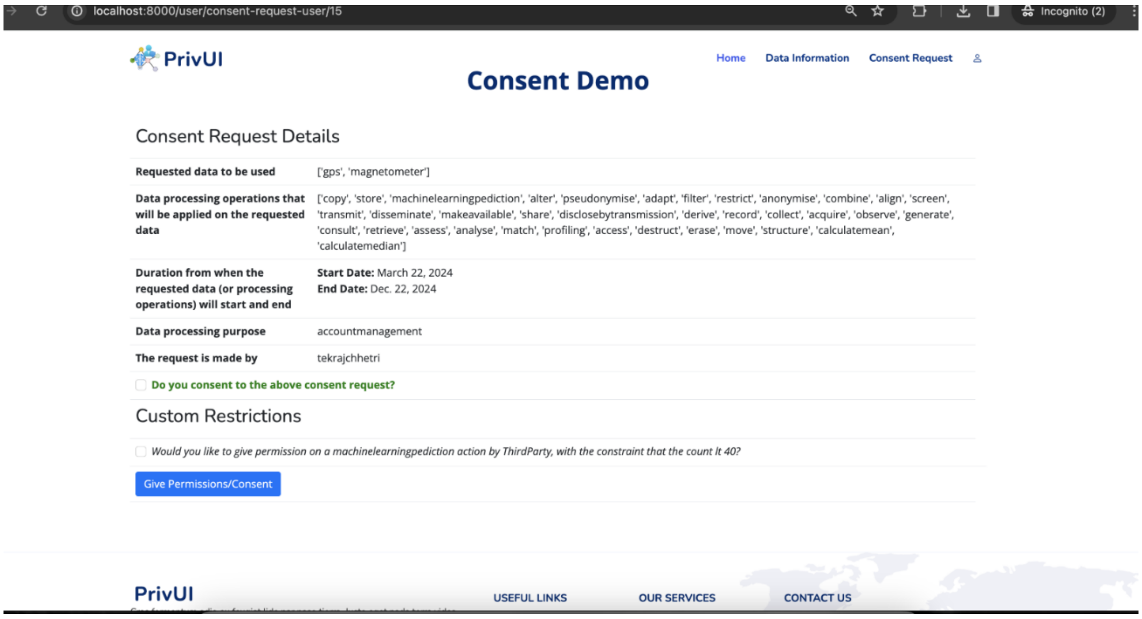
Consent Request

Request Form	Consent Status	Response Status	Action
UPCAST project demo consent request	NotGiven	REVOKED	View Response
[redacted]	[redacted]	[redacted]	
Consent Demo	Requested	REQUESTED	Grant Consent

PrivUI
Cras fermentum odio eu feugiat lide par naso tierra. Justo eget nada terra videa magna derita valies darta donna mare fermentum iaculis eu non diam phasellus.

USEFUL LINKS	OUR SERVICES	CONTACT US
Home	Web Design	A108 Adam Street
About us	Web Development	New York, NY 535022
Services	Product Management	United States
Terms of service	Marketing	**Phone:** +1 5589 55488 55
Privacy policy	Graphic Design	**Email:** info@example.com

Figure 12: Consent request received by data producer and response status



Consent Demo

Consent Request Details

Requested data to be used	['gps', 'magnetometer']
Data processing operations that will be applied on the requested data	['copy', 'store', 'machinelearningpediction', 'alter', 'pseudonymise', 'adapt', 'filter', 'restrict', 'anonymise', 'combine', 'align', 'screen', 'transmit', 'disseminate', 'makeavailable', 'share', 'disclosebytransmission', 'derive', 'record', 'collect', 'acquire', 'observe', 'generate', 'consult', 'retrieve', 'assess', 'analyse', 'match', 'profiling', 'access', 'destruct', 'erase', 'move', 'structure', 'calculatemean', 'calculatemedian']
Duration from when the requested data (or processing operations) will start and end	**Start Date:** March 22, 2024 **End Date:** Dec. 22, 2024
Data processing purpose	accountmanagement
The request is made by	tekrajchhetri

☐ **Do you consent to the above consent request?**

Custom Restrictions

☐ *Would you like to give permission on a machinelearningpediction action by ThirdParty, with the constraint that the count lt 40?*

Give Permissions/Consent

Figure 13: Consent along with the use case specific constraints details and the opt-in opt-out option to grant (or prohibit) the permission (or consent)

2.1.3 Data Consumer internal policies and Data Processing Workflows:

The following three capabilities are demonstrated:

1) Specification of an organisation's internal policies that regulate its operations against a variety of provisions; in the general case, these may originate from applicable legislation (e.g., the GDPR) or other operational and data governance considerations. Notably, the same technology can be used to accommodate, by extension, policies governing entities in established collaborations, as defined, e.g., on behalf of a marketplace.
2) Data Processing Workflow (DPW) specification, as the primary means through which the data consumer states their intentions for the data they seek to acquire. These intentions are derived from jointly considering a variety of aspects, including: the processing operations intended to be performed; the entities in direct or indirect control of their execution; the attributes of the acquired data and the conditions under which any processing and exchange is meant to take place; the stated purposes that the DPW in question is intended to serve.
3) Conflict identification and resolution. This is currently demonstrated at the level of the organisation, in the context of addressing discrepancies between any user-defined DPW and the organisation's internal policies. At a second phase, this functionality will be further exploited in the context of the Negotiation plugin, towards resolving the conflicts that may arise between a data consumer-defined DPW and the constraints specified by data provider(s) using the functionalities described in sections 2.3.1 and 2.3.2.

Management of Resource Consumer-side policies and DPWs is demonstrated via the goodFlows tool which offers the required functionalities by means of dedicated user interfaces, i.e., Policy Administration and Process Modeller, respectively.

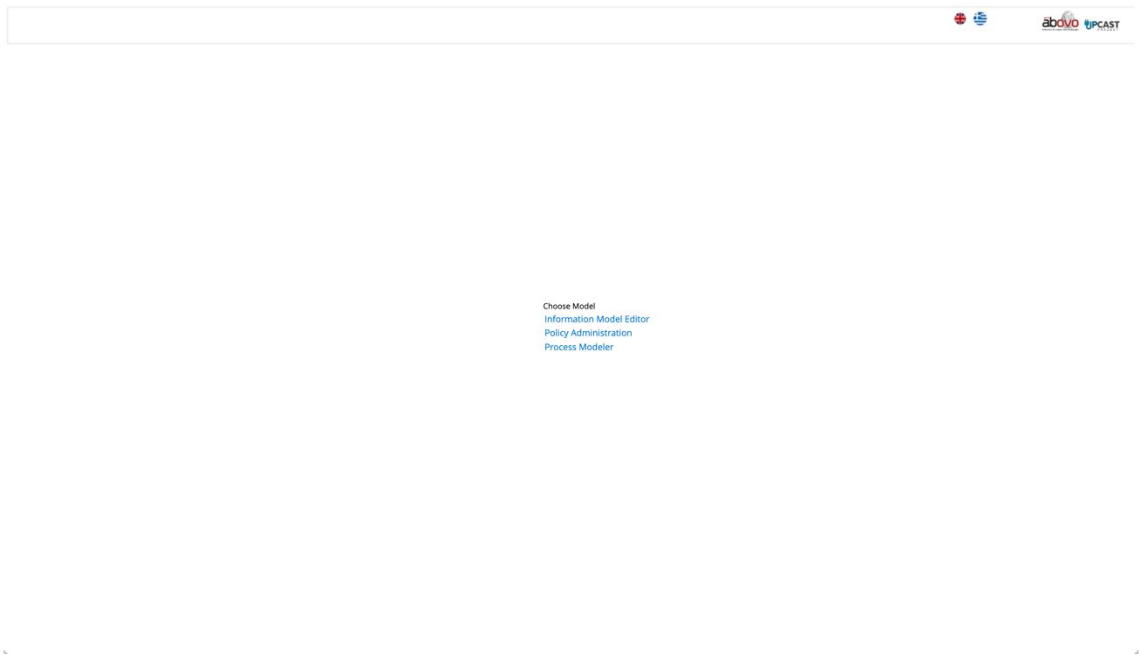
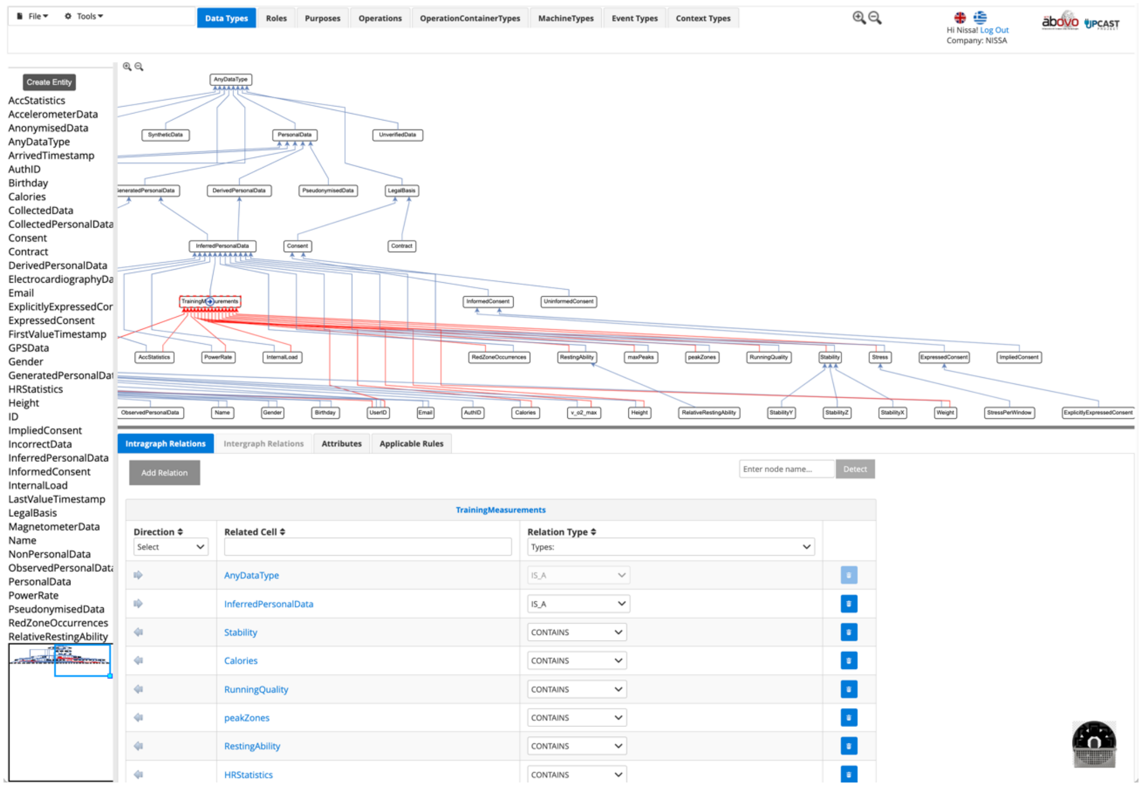


Figure 14: GoodFlows homepage

Both policies (internal organisational – domain specific policies, policies stemming from the legislation, or imposed by an associated marketplace) and DPWs are defined on the basis of the organisation's Information Model. To this end, the goodFlows Information Model Editor view is leveraged, i.e., a user-friendly software application devised for the specification and management of the underlying semantic model. It provides all necessary functionality for the management of all different information classes and their relations, and the corresponding ontology. The use of the editor hides the technical details of the model, requiring no particular technical expertise by its users; it translates the input provided through the graphical interface to machine code. As illustrated in Figure 15, there are three main areas the user sees upon logging in, notably the Graph (upper side of the screen), List (left side of the screen) and Properties (bottom side of the screen) areas. The Graph area allows graphically editing the different information classes, including the definition of entities and the hierarchies thereof. As shown in the upper part of this area, it is organised in tabs, each used for the illustration of a different graph of the Information Model. The List area provides a list of the entities, sorted alphabetically, in order to facilitate navigation. The Properties area is devised for the management of entities' properties, notably attributes and relations, both inside the same graph (i.e., hierarchy) and relations spanning across different graphs (e.g., the association of an operation with sought purposes), as well as providing information about applicable rules for the selected entity. The user interface is complemented by some control functions, as well as a map for making easier the navigation in large graphs.

Setting up the organisational Information Model

For this demo, a domain-specific information model has been devised modelling data types, purposes, operations, etc., met in the operation of the Health and Fitness use case by partner NISSA. In that respect, it is established upon the Data Privacy Vocabulary (DPV), while NISSA specific concepts fall under high-level concepts by means of `isA` relations. As an example, Figure 15 illustrates (part of) the DataTypes' graph of a model. As shown, there are multiple entities in the graph, interconnected with two types of relations; blue lines denote `isA` relation, whereas red lines reflect the `isPartOf` relation. Since the entity selected by the user is the `TrainingMeasurements`, the Properties area lists the ones characterising this entity. To this end, there are shown various `contains` relations (with Calories, `peakZones`, `RestingAbility`, `UserID`, `Height`, `Weight` etc.), as well as `isA` relations, with `InferredPersonalData` and with `AnyDataType` being the most general data type. Likewise, NISSA domain-specific entities have been added in (i) the Operations' graph (Figure 16), e.g., the operations `PerformStatisticalAnalysis` (`isA Analyse`), `CalculateCaloriesBurnt` (`isPartOf PerformStatisticalAnalysis`), `PublishResults` (`isA MakeAvailable`) and `ProvideConsent` (`isA anyOperation`); (ii) the Roles' graph (Figure 17), e.g., `DataAnalyst` (`isA NaturalPerson`), `Trainer` (`isA User`) and `Trainee` (`isA User`); (iii) the Purposes' graph (Figure 18), e.g., `Estimate_fitness_level` (`isA PersonalisedBenefits`) and `Leaderboard` (`isA RequestedServiceProvision`).



AccStatistics
AccelerometerData
AnonymisedData
AnyDataType
ArrivedTimestamp
AuthID
Birthday
Calories
CollectedData
CollectedPersonalData
Consent
Contract
DerivedPersonalData
ElectrocardiographyDa
Email
ExplicitlyExpressedCor
ExpressedConsent
FirstValueTimestamp
GPSData
Gender
GeneratedPersonalDat
HRStatistics
Height
ID
ImpliedConsent
IncorrectData
InferredPersonalData
InformedConsent
InternalLoad
LastValueTimestamp
LegalBasis
MagnetometerData
Name
NonPersonalData
ObservedPersonalData
PersonalData
PowerRate
PseudonymisedData
RedZoneOccurrences
RelativeRestingAbility

TrainingMeasurements

Direction	Related Cell	Relation Type	
Select		Types:	
→	AnyDataType	IS_A	
→	InferredPersonalData	IS_A	
←	Stability	CONTAINS	
←	Calories	CONTAINS	
←	RunningQuality	CONTAINS	
←	peakZones	CONTAINS	
←	RestingAbility	CONTAINS	
←	HRStatistics	CONTAINS	

Figure 15: Information Model Editor showing DataTypes' graph

The formed hierarchies allow for defining rules upon high-level entities which will be propagated across the hierarchies, alleviating the need to explicitly define exhaustive rulesets. In that respect, both positive and negative authorisations will be propagated to more specific entities, while negative authorisations will also be propagated across `isPartOf` relations; for instance, a permission defined for the role `DataSubject` will be inherited towards `Trainee`, while a prohibition defined for data type `UserID` will be propagated to the container `TrainingMeasurements` entity, thus implying partial access to the said entity.

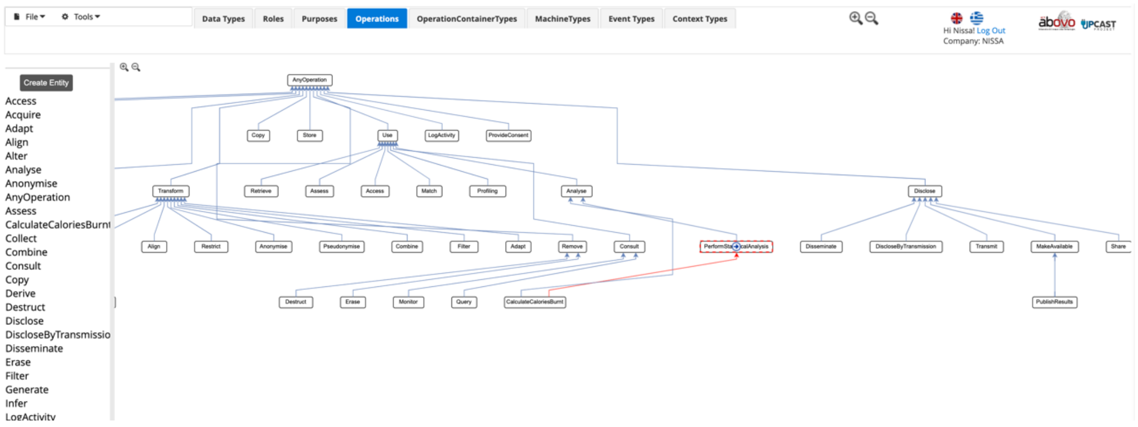


Figure 16: Operations graph

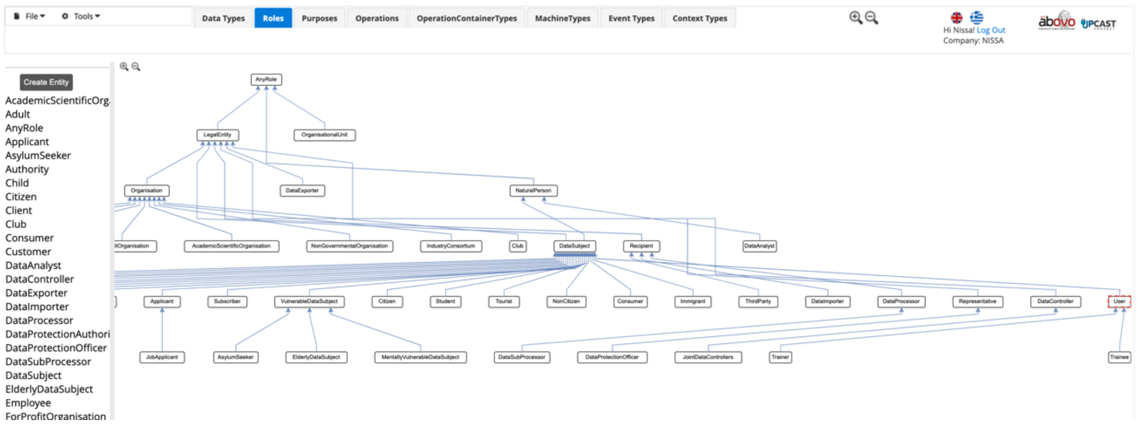


Figure 17: Roles graph

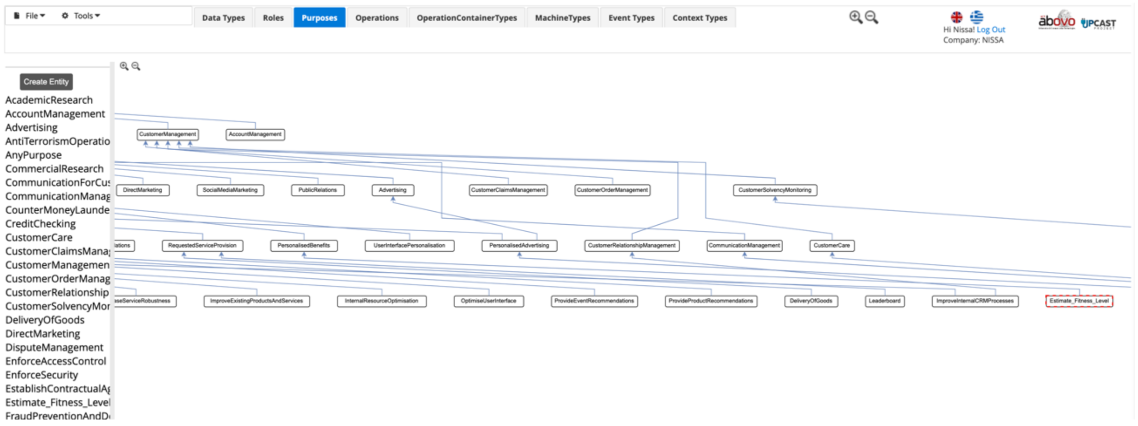


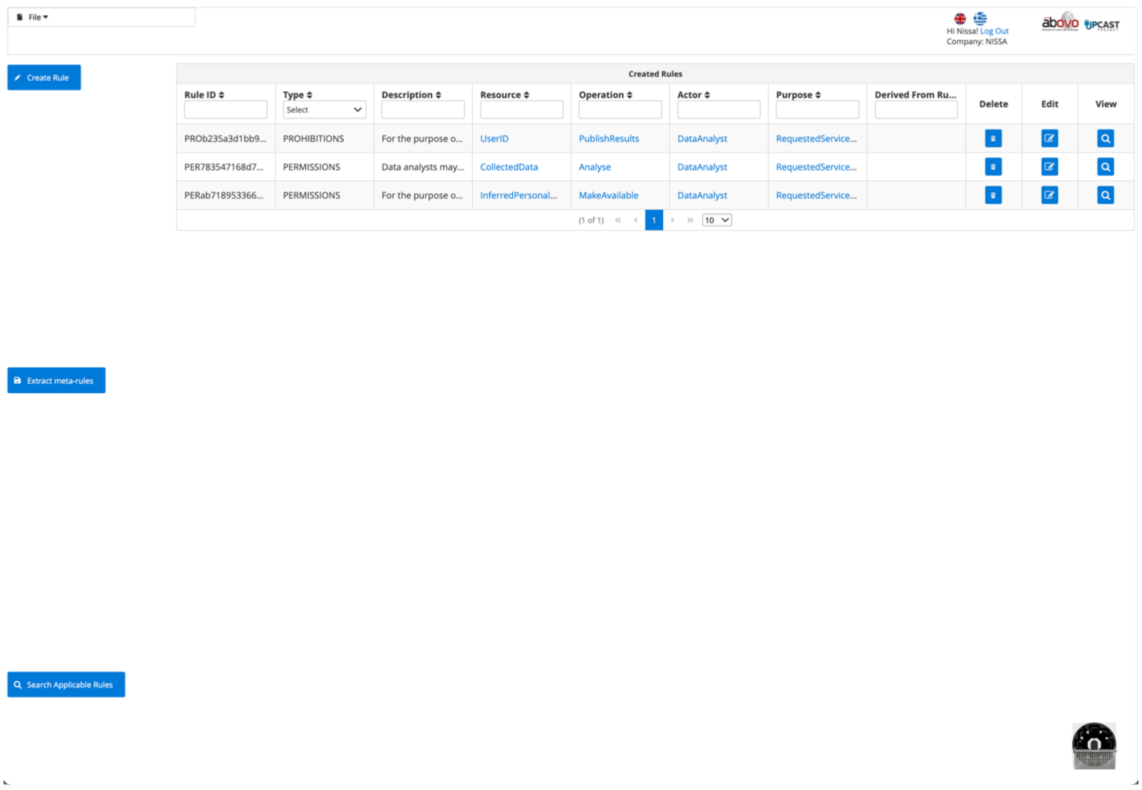
Figure 18: Purposes graph

Policy Administration

The Policy Administration editor is integrated with the one for Information Model administration, as rules are defined upon its entities. The editor provides functionality for creating, editing, deleting and listing rules, as well as extracting meta-rules from explicitly defined ones for high-level entities and rules' searching by means of advanced search criteria, while it guides the user through the definition of a rule's elements in a user-friendly manner. Said functionalities are made available to the UI through the appropriate REST API depicted in Annex 2. The use of the UI hides the technical details of the underlying semantic policy model, requiring no particular technical expertise by its users.

1. View and manage access and usage control rules

Figure 19 depicts the home page of the Policy Model Editor. Specified rules are listed in a table and can be viewed, edited and deleted using the respective row buttons. The "Create Rule" button allows for the creation of new rules, while the button "Extract-metarules" initiates the procedure that generates the meta-rules being the result of propagating explicitly defined rules across the information model's hierarchies; this button has been added for testing purposes, as the related functionality will be finally offered by a respective event-driven job executed every time the policy model has been finalised after being updated.



Created Rules

Rule ID	Type	Description	Resource	Operation	Actor	Purpose	Derived From Ru...	Delete	Edit	View
PROb235a3d1bb9...	PROHIBITIONS	For the purpose o...	UserID	PublishResults	DataAnalyst	RequestedService...				
PER783547168d7...	PERMISSIONS	Data analysts may...	CollectedData	Analyse	DataAnalyst	RequestedService...				
PERab718953366...	PERMISSIONS	For the purpose o...	InferredPersonal...	MakeAvailable	DataAnalyst	RequestedService...				

(1 of 1) 1 10

Figure 19: Policy administration Home view

It is noted that the system is Deny-biased, i.e. something is permitted if at least one associated permission has been defined.

2. Create access and usage control rules

In goodFlows, an access and usage control rule is modelled through the following structure:

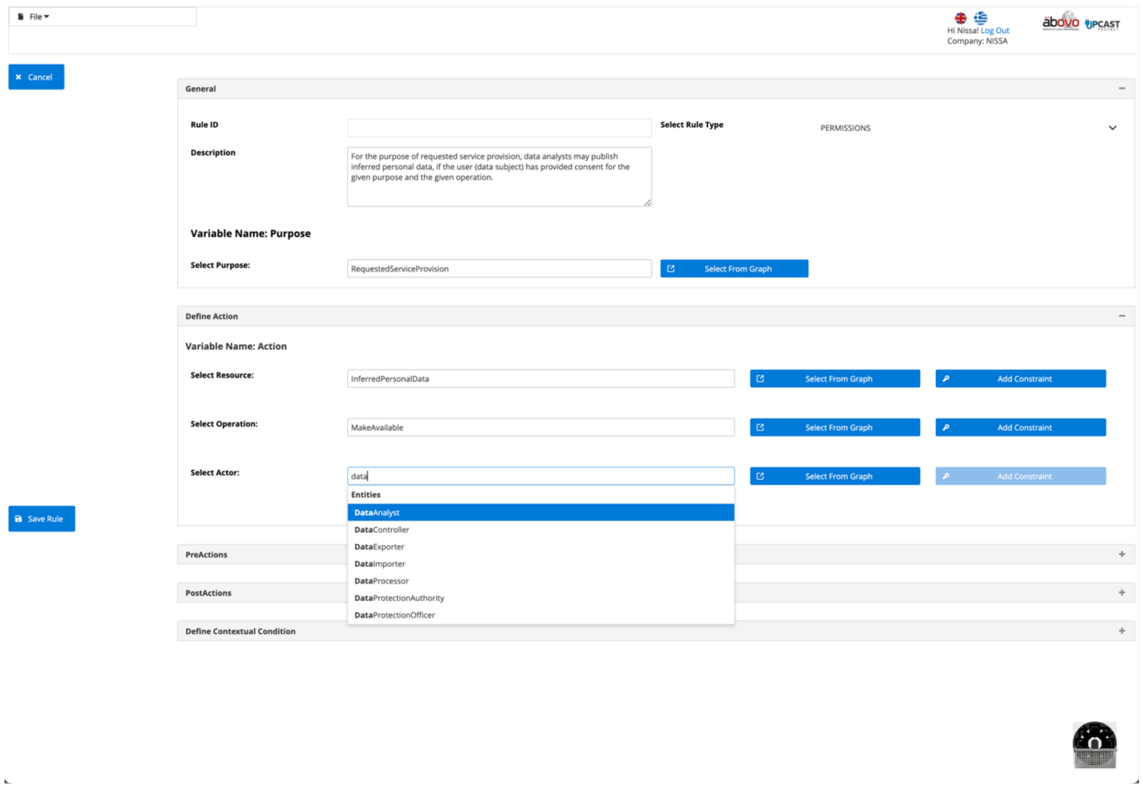
$$\left.\begin{matrix} Permission \\ Prohibition \\ Obligation \end{matrix}\right\} (pu, act, preAct, cont, postAct)$$

where *act* is the action that the rule applies to; *pu* is the purpose for which *act* is permitted/prohibited/obliged to be executed; *cont* is a structure of contextual parameters; *preAct* is a structure of actions that should have preceded; *postAct* refers to the action(s) that must be executed following the rule enforcement.

Actions constitute conceptual triples of *{actor, operation, resource}* and are formed leveraging the entities of the Information Model. In more detail, action's entities are defined as a tuple <*concept*, *constraint*>, where concept is an Information Model entity, and constraint is an expression (or a logical relation thereof) assigning values to attributes and/or sub-concepts of the given entity. This mechanism allows the definition of access and usage control rules inline with the Attribute-based Access Control (ABAC) paradigm.

Authoring of such expressive but complex rules is a very challenging task, implying the need for a trade-off between expressiveness and user friendliness; the Rule Creation form illustrated in Figure 20 has been designed with a view to providing an intuitive way of specifying fine-grained access and usage control rules. The form is divided in the following five areas:

- The area for defining the rule's general information, i.e., the rule type (Permission, Prohibition or Obligation), a brief description of the rule and the purpose for which this rule applies.
- The access and usage action definition area, for defining the elements of the action for which the rule applies.
- The pre-actions definition area, for defining one or more (logically related) required actions that must or must not have preceded the enforcement of the rule.
- The post-actions definition area, for defining one or more (logically related) required actions that must or must not follow the enforcement of the rule.
- The contextual constraints definition area for defining one or more (logically related) contextual conditions under which the rule applies.



General

Rule ID:

Select Rule Type: PERMISSIONS

Description: For the purpose of requested service provision, data analysts may publish inferred personal data, if the user (data subject) has provided consent for the given purpose and the given operation.

Variable Name: Purpose

Select Purpose: RequestedServiceProvision

Define Action

Variable Name: Action

Select Resource: InferredPersonalData

Select Operation: MakeAvailable

Select Actor: data

Entities
- **Data**Analyst
- **Data**Controller
- **Data**Exporter
- **Data**Importer
- **Data**Processor
- **Data**ProtectionAuthority
- **Data**ProtectionOfficer

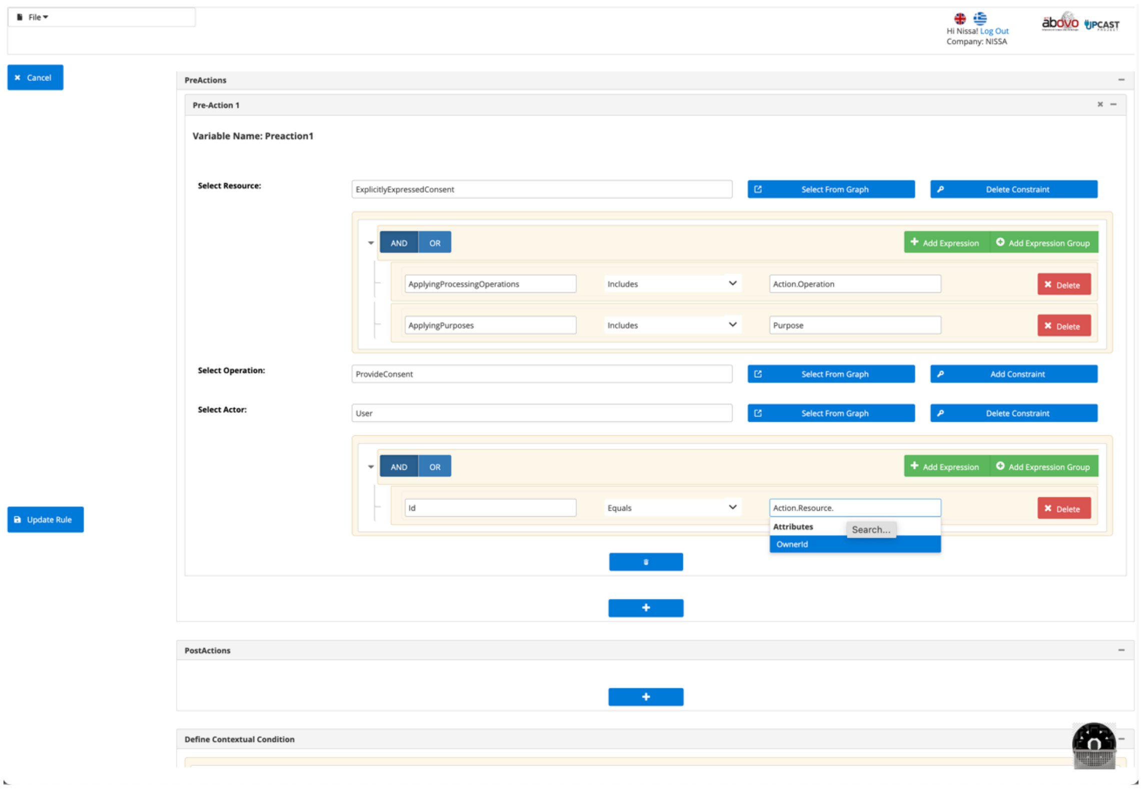
Figure 20: Rule Creation form – General rule information and definition of action for which the rule applies

Information coming from the Information Model Editor, ~~like~~ e.g., purposes, data types, operations, roles and attributes, become available either through pop-up dialogs (by pressing the "Select From Graph" button) or by typing in the respective textboxes with suggestions made by the system through autocompletion.

Various variables are defined for rule's elements by the system, in order to be used in the context of other entities' definition; specifically, based on the label of the referenced entity, the following variables are available:

- Access and usage action specific: {Action}, {Action}.Resource, {Action}.Operation, {Action}.Actor
- Pre-action specific: {Preaction<id>}, {Preaction<id>}.Resource, {Preaction<id>}.Operation, {Preaction<id>}.Actor,
- Post-action specific: {Postaction<id>}, {Postaction<id>}.Resource, {Postaction<id>}.Operation, {Postaction<id>}.Actor
- Purpose
- Context

For instance, Figure 21 illustrates an example of defining the preAction of the permission denoting "For the purpose of requested service provision, data analysts may publish inferred personal data, if the user (data subject) has provided consent for the given purpose and the given operation." Logically related constraints are defined for the resource `ExplicitlyExpressedConsent`, leveraging the variables Purpose and Action.Operation, while the User is associated with the owner of the main action's resource (i.e., `InferredPersonalData`).



PreActions

Pre-Action 1

Variable Name: Preaction1

Select Resource: ExplicitlyExpressedConsent

ApplyingProcessingOperations Includes Action.Operation

ApplyingPurposes Includes Purpose

Select Operation: ProvideConsent

Select Actor: User

Id Equals Action.Resource.

Attributes

OwnerId

PostActions

Define Contextual Condition

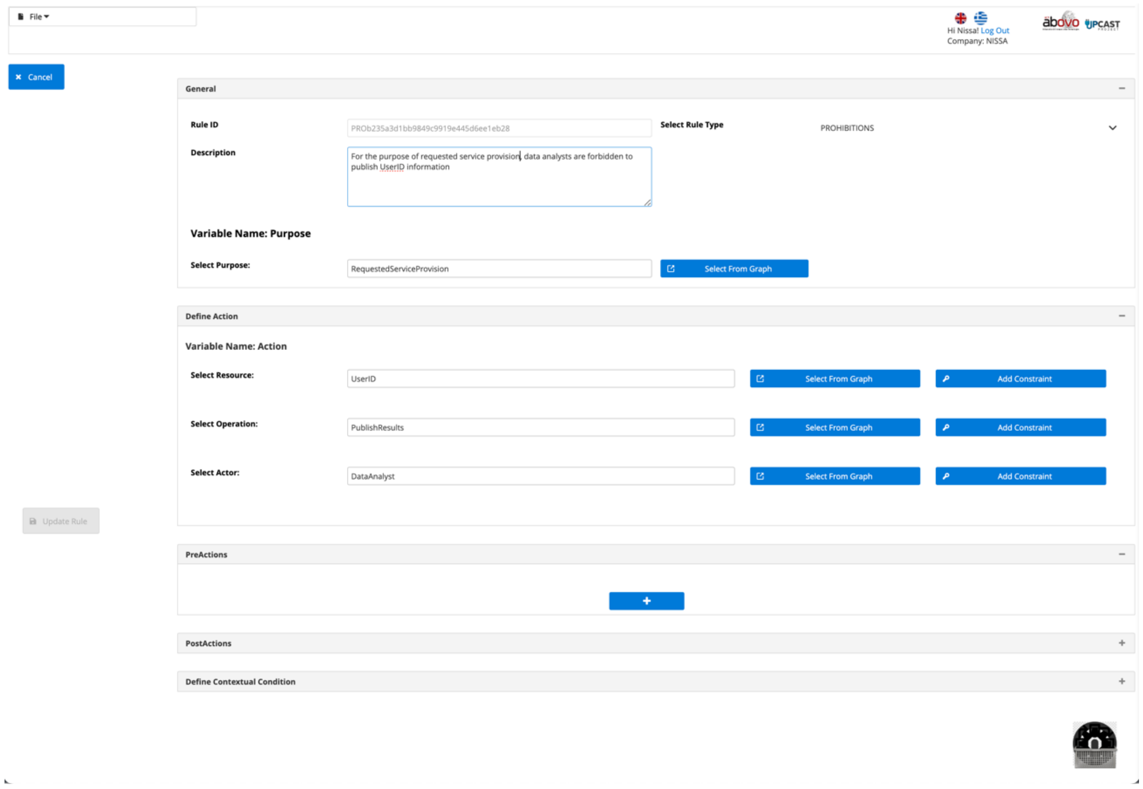
Figure 21: Rule Creation form - defining preActions and constraints upon action entities

Regarding the definition of constraints, autocompletion facilitates the definition of complex expressions regarding the constraint subject and value, drawing inspiration from SQL; for instance, adding a period character after the "Action.Resource" of Figure 22 will fetch the attributes of the `InferredPersonalData` dataType.

Finally, if more than one pre-/post- actions are specified, they must be logically related to each other before saving the rule; logical relations of required actions are defined with the same component used for the definition of logical sentences of constraints.

Once they have created or modified a rule, the user may choose to save it through the corresponding button at the bottom left of the main screen, named after "Save Rule" or "Update Rule" respectively. Rules are saved in the underlying RDF Triplestore.

Further, the prohibition of publishing `UserID` for the purpose of `RequestedServiceProvision` is defined. This rule will act as an exception to the previously defined permission, as it will be propagated across the `isPartOf` relation between `UserID` and `TrainingMeasurements`.


Hi Nissa! Log Out
Company: NISSA

General

Rule ID: PROb235a3d1bb9849c9919e445d6ee1eb28

Select Rule Type: PROHIBITIONS

Description: For the purpose of requested service provision, data analysts are forbidden to publish UserID information

Variable Name: Purpose

Select Purpose: RequestedServiceProvision

Define Action

Variable Name: Action

Select Resource: UserID

Select Operation: PublishResults

Select Actor: DataAnalyst

PreActions

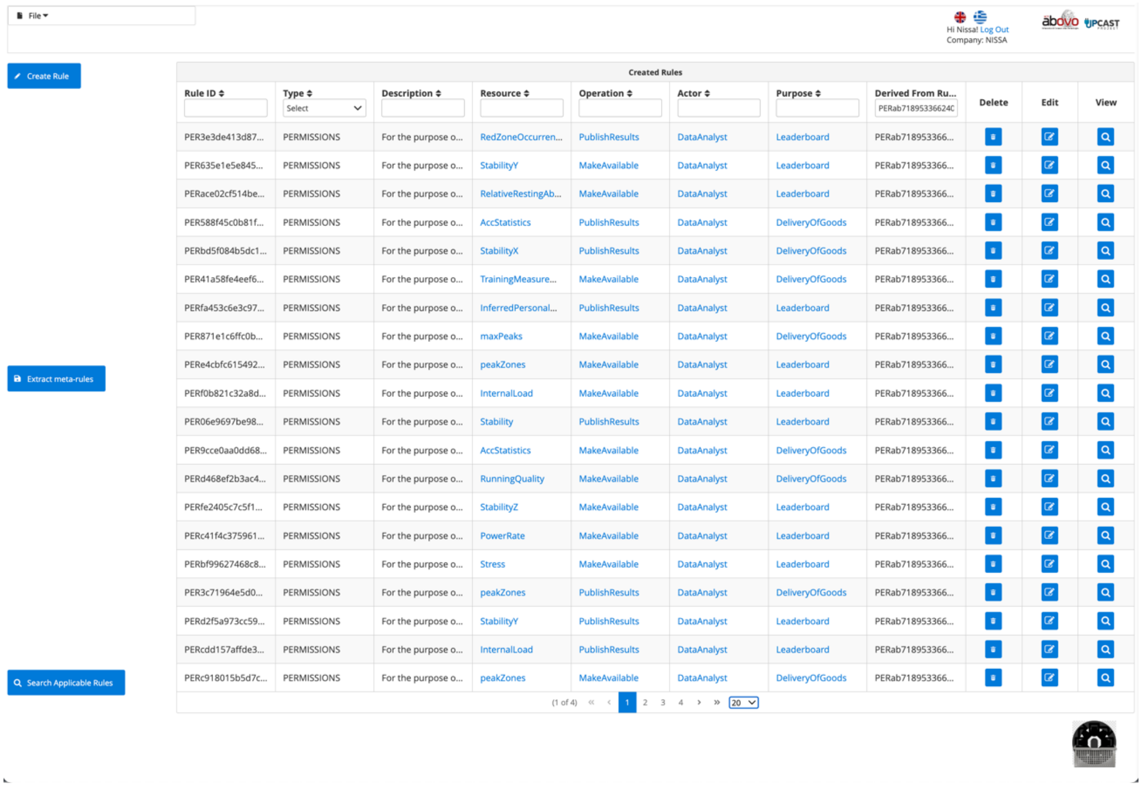
PostActions

Define Contextual Condition

Figure 22: Example of prohibition regarding publishing an UserID

3. Extract meta-rules

Motivated by the need for improved performance during real-time operation, such as the conflict identification and resolution in the context of validating a DPW against the access and usage control rules, offline reasoning, i.e., proactive extraction of knowledge contained in the access and usage control rules, is leveraged. Through the meta-rules extraction procedure, most of the required knowledge will be already available by the request time, thus offering performance gains. Once the ruleset is finalised, meta-rules are extracted by pressing the “Extract meta-rules” button through propagation of the explicitly defined rules across the Information Model’s graphs.



File

Hi Nissa! Log Out
Company: NISSA

Create Rule

Extract meta-rules

Search Applicable Rules

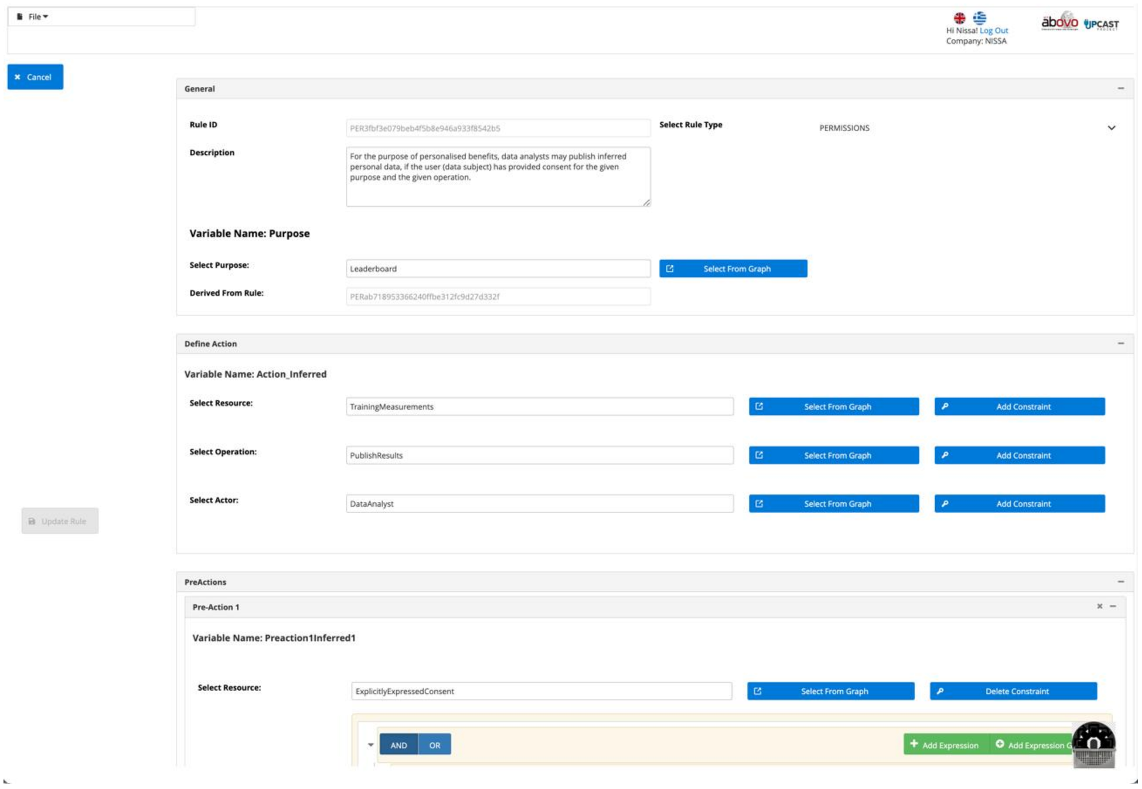
Created Rules

Rule ID	Type	Description	Resource	Operation	Actor	Purpose	Derived From Ru...	Delete	Edit	View
	Select						PERab71895336624C			
PER3e3de413d87...	PERMISSIONS	For the purpose o...	RedZoneOccurren...	PublishResults	DataAnalyst	Leaderboard	PERab718953366...	🗑	✎	🔍
PER635e1e5e845...	PERMISSIONS	For the purpose o...	StabilityY	MakeAvailable	DataAnalyst	Leaderboard	PERab718953366...	🗑	✎	🔍
PERace02cf514be...	PERMISSIONS	For the purpose o...	RelativeRestingAb...	MakeAvailable	DataAnalyst	Leaderboard	PERab718953366...	🗑	✎	🔍
PER588f45c0b81f...	PERMISSIONS	For the purpose o...	AccStatistics	PublishResults	DataAnalyst	DeliveryOfGoods	PERab718953366...	🗑	✎	🔍
PERbd5f084b5dc1...	PERMISSIONS	For the purpose o...	StabilityX	PublishResults	DataAnalyst	DeliveryOfGoods	PERab718953366...	🗑	✎	🔍
PER41a58fe4eef6...	PERMISSIONS	For the purpose o...	TrainingMeasure...	MakeAvailable	DataAnalyst	DeliveryOfGoods	PERab718953366...	🗑	✎	🔍
PERfa453c6e3c97...	PERMISSIONS	For the purpose o...	InferredPersonal...	PublishResults	DataAnalyst	Leaderboard	PERab718953366...	🗑	✎	🔍
PER871e1c6ffc0b...	PERMISSIONS	For the purpose o...	maxPeaks	MakeAvailable	DataAnalyst	DeliveryOfGoods	PERab718953366...	🗑	✎	🔍
PERe4cbfc615492...	PERMISSIONS	For the purpose o...	peakZones	MakeAvailable	DataAnalyst	Leaderboard	PERab718953366...	🗑	✎	🔍
PERf0b821c32a8d...	PERMISSIONS	For the purpose o...	InternalLoad	MakeAvailable	DataAnalyst	Leaderboard	PERab718953366...	🗑	✎	🔍
PER06e9697be98...	PERMISSIONS	For the purpose o...	Stability	PublishResults	DataAnalyst	Leaderboard	PERab718953366...	🗑	✎	🔍
PER9cce0aa0dd68...	PERMISSIONS	For the purpose o...	AccStatistics	MakeAvailable	DataAnalyst	DeliveryOfGoods	PERab718953366...	🗑	✎	🔍
PERd468ef2b3ac4...	PERMISSIONS	For the purpose o...	RunningQuality	MakeAvailable	DataAnalyst	DeliveryOfGoods	PERab718953366...	🗑	✎	🔍
PERfe2405c7c5f1...	PERMISSIONS	For the purpose o...	StabilityZ	MakeAvailable	DataAnalyst	Leaderboard	PERab718953366...	🗑	✎	🔍
PERc41f4c375961...	PERMISSIONS	For the purpose o...	PowerRate	MakeAvailable	DataAnalyst	Leaderboard	PERab718953366...	🗑	✎	🔍
PERbf99627468c8...	PERMISSIONS	For the purpose o...	Stress	MakeAvailable	DataAnalyst	Leaderboard	PERab718953366...	🗑	✎	🔍
PER3c71964e5d0...	PERMISSIONS	For the purpose o...	peakZones	PublishResults	DataAnalyst	DeliveryOfGoods	PERab718953366...	🗑	✎	🔍
PERd2f5a973cc59...	PERMISSIONS	For the purpose o...	StabilityY	PublishResults	DataAnalyst	Leaderboard	PERab718953366...	🗑	✎	🔍
PERcdd157affde3...	PERMISSIONS	For the purpose o...	InternalLoad	PublishResults	DataAnalyst	Leaderboard	PERab718953366...	🗑	✎	🔍
PERc918015b5d7c...	PERMISSIONS	For the purpose o...	peakZones	MakeAvailable	DataAnalyst	DeliveryOfGoods	PERab718953366...	🗑	✎	🔍

(1 of 4) « ‹ 1 2 3 4 › » 20

Figure 23: Meta-rules generated from the permissions rule in Figures 19 and 20

For instance, Figure 24 illustrates some of the meta-rules generated from the rule defined in Figure 22 and Figure 23, while Figure 25 and Figure 26 present the details of such a meta-rule applying for the purpose `Leaderboard`, the resource `TrainingMeasurements` and the operation `PublishResults`. Specifically, Figure 15 and Figure 16 present the analysis of the explicitly defined preAction to two OR-related preActions, depending on whether the `User` is a `Trainer` or a `Trainee`.



General

Rule ID: PER3fbf3e079beb4f5b8e946a933f8542b5

Select Rule Type: PERMISSIONS

Description: For the purpose of personalised benefits, data analysts may publish inferred personal data, if the user (data subject) has provided consent for the given purpose and the given operation.

Variable Name: Purpose

Select Purpose: Leaderboard

Derived From Rule: PERab718953366240ffbe312fc9d27d332f

Define Action

Variable Name: Action_Inferred

Select Resource: TrainingMeasurements

Select Operation: PublishResults

Select Actor: DataAnalyst

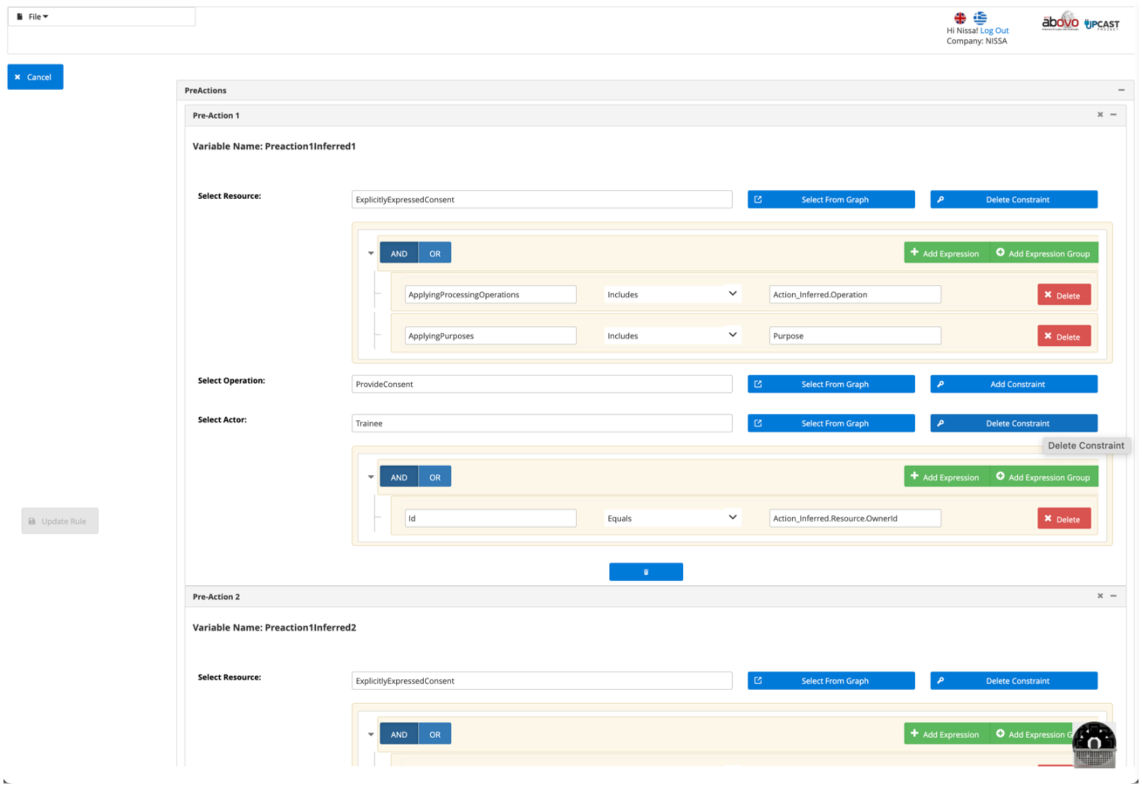
PreActions

Pre-Action 1

Variable Name: Preaction1Inferred1

Select Resource: ExplicitlyExpressedConsent

Figure 24: Meta-rule details (a)



PreActions

Pre-Action 1

Variable Name: Preaction1Inferred1

Select Resource: ExplicitlyExpressedConsent

ApplyingProcessingOperations	Includes	Action_Inferred.Operation
ApplyingPurposes	Includes	Purpose

Select Operation: ProvideConsent

Select Actor: Trainee

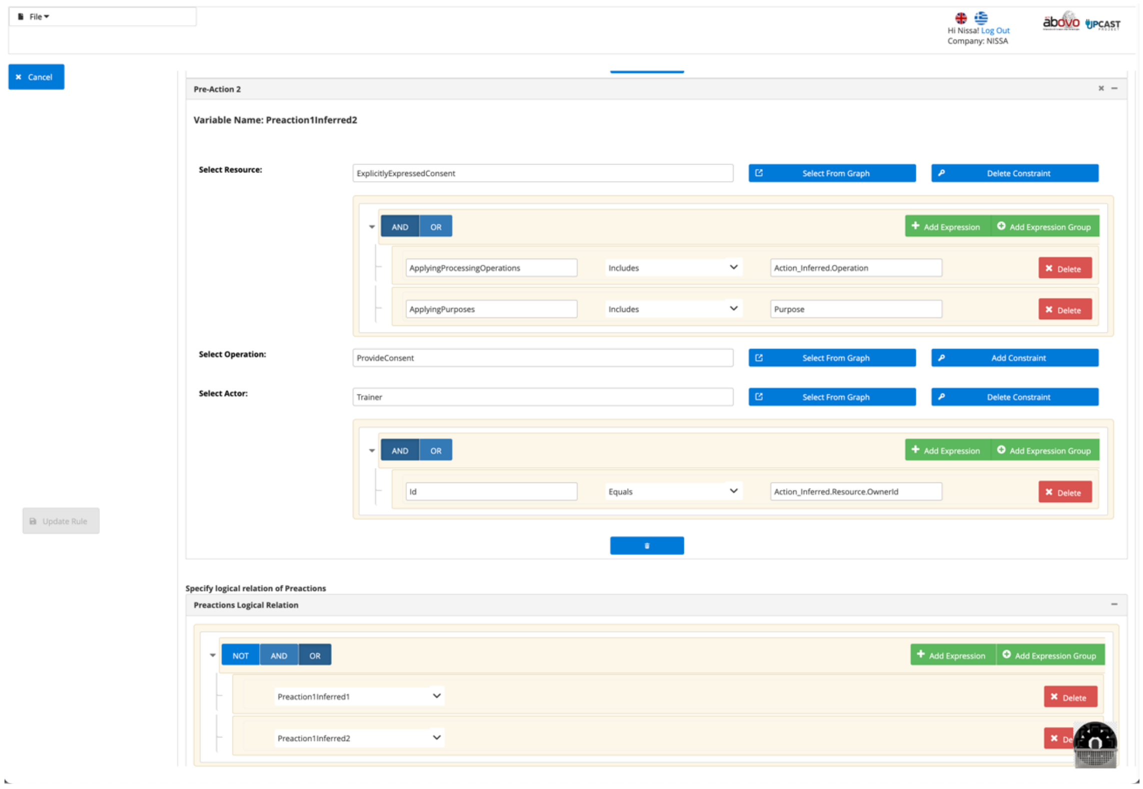
Id	Equals	Action_Inferred.Resource.OwnerId

Pre-Action 2

Variable Name: Preaction1Inferred2

Select Resource: ExplicitlyExpressedConsent

Figure 25: Meta-rule details (b)



Pre-Action 2

Variable Name: Preaction1Inferred2

Select Resource: ExplicitlyExpressedConsent

ApplyingProcessingOperations	Includes	Action_Inferred.Operation
ApplyingPurposes	Includes	Purpose

Select Operation: ProvideConsent

Select Actor: Trainer

Id	Equals	Action_Inferred.Resource.OwnerId

Specify logical relation of Preactions

Preactions Logical Relation

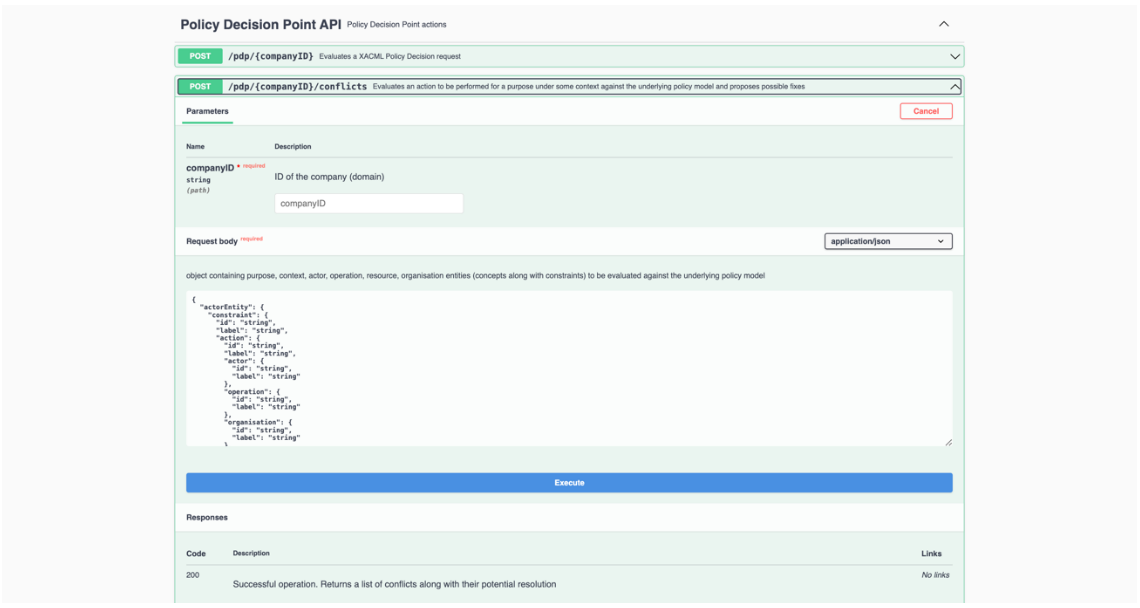
Preaction1Inferred1

Preaction1Inferred2

Figure 26: Meta-rule details (c)

Policy decision

Reasoning over the access and usage control rules will feed the procedure of DPW verification and re-engineering, towards compliant by design DPWs and efficient negotiation between Data Consumers and Data Providers. For this purpose and fostering explainability, the Policy Decision Point API (Figure 27) offers functionality for evaluating a “standalone intention”, i.e., an attempted action for a purpose and under some context against the underlying ruleset in order to identify conflicts and to propose alternatives.


Policy Decision Point API Policy Decision Point actions

POST /pdp/{companyID} Evaluates a XACML Policy Decision request

POST /pdp/{companyID}/conflicts Evaluates an action to be performed for a purpose under some context against the underlying policy model and proposes possible fixes

Parameters

Name	Description
companyID * required string *(path)*	ID of the company (domain) companyID

Request body required — application/json

object containing purpose, context, actor, operation, resource, organisation entities (concepts along with constraints) to be evaluated against the underlying policy model

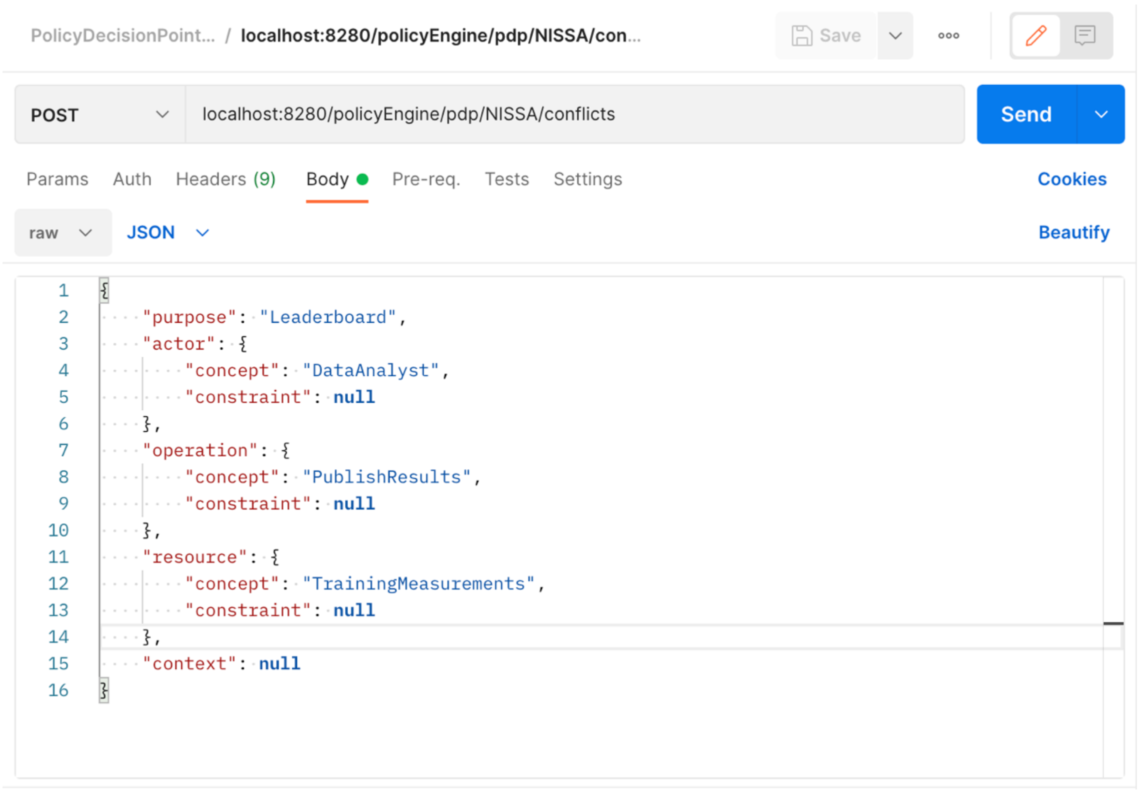
```
{
  "actorEntity": {
    "constraint": {
      "id": "string",
      "label": "string",
      "action": {
        "id": "string",
        "label": "string",
        "actor": {
          "id": "string",
          "label": "string"
        },
        "operation": {
          "id": "string",
          "label": "string"
        },
        "organisation": {
          "id": "string",
          "label": "string"
        }
```

Responses

Code	Description	Links
200	Successful operation. Returns a list of conflicts along with their potential resolution	*No links*

Figure 27: Policy Decision Point API – Conflict identification and possible resolution

In the context of this demo, the standalone intention of Figure 28 will be evaluated against the rules defined in the previous Policy Administration steps for NISSA, namely whether a data analyst may publish training measurements for the purpose of projection to the leaderboard. It is noted that this functionality is not yet offered via the UI.

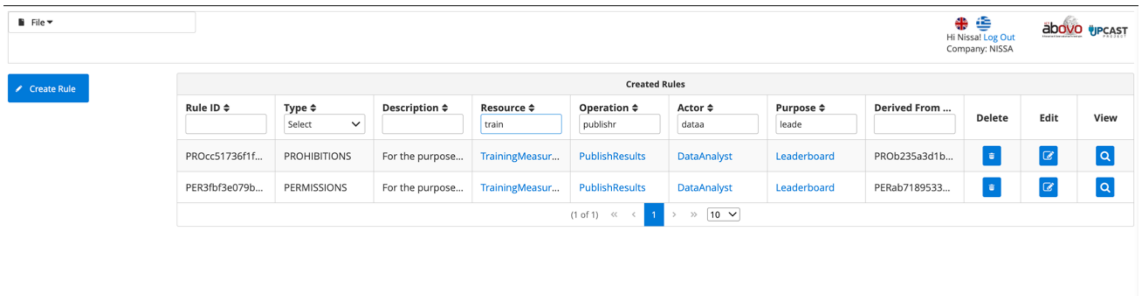

PolicyDecisionPoint... / localhost:8280/policyEngine/pdp/NISSA/con...

POST localhost:8280/policyEngine/pdp/NISSA/conflicts

```
{
    "purpose": "Leaderboard",
    "actor": {
        "concept": "DataAnalyst",
        "constraint": null
    },
    "operation": {
        "concept": "PublishResults",
        "constraint": null
    },
    "resource": {
        "concept": "TrainingMeasurements",
        "constraint": null
    },
    "context": null
}
```

Figure 28: Evaluate standalone intention

After the meta-rules extraction procedure, the two rules found after search of Figure 19 may apply for the given intention, with their details illustrated in Figure 14, Figure 15, Figure 16 and Figure 20. However, the found prohibition prevails over the permission and subsequently the system will examine if partial access to/usage of the requested resource is allowed.

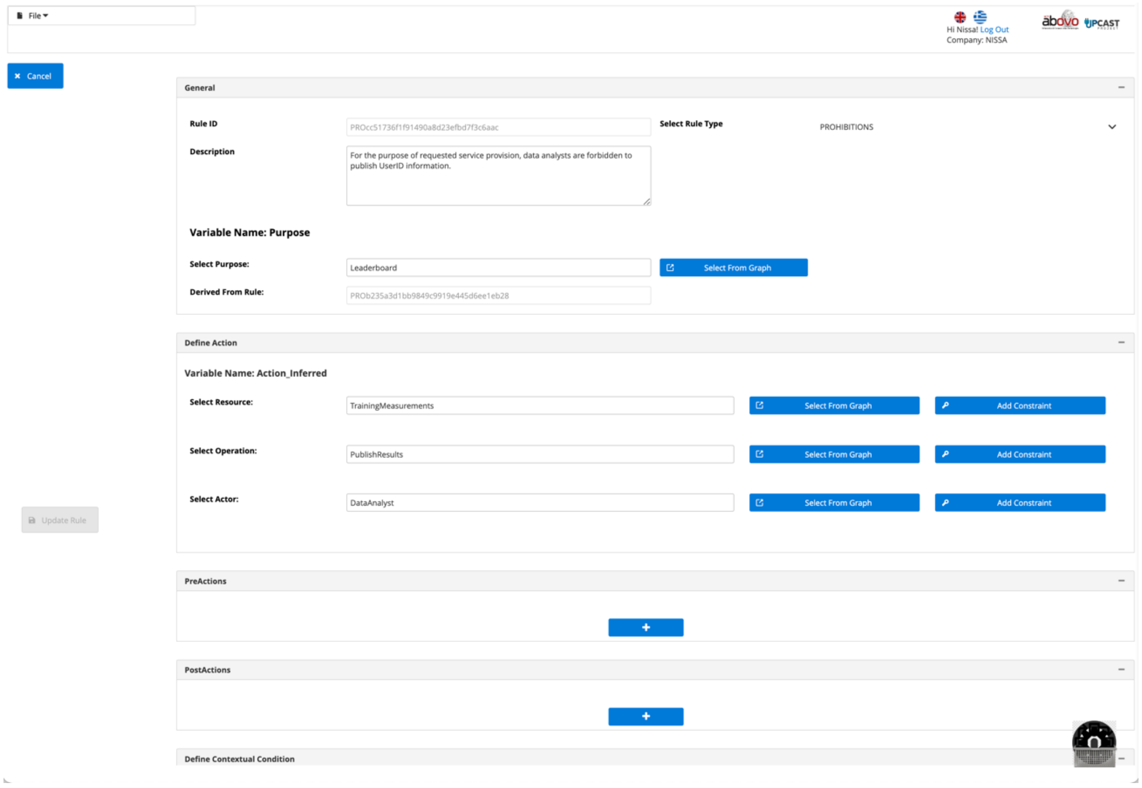

Hi Nissa! Log Out
Company: NISSA

Created Rules

Rule ID	Type	Description	Resource	Operation	Actor	Purpose	Derived From ...	Delete	Edit	View
	Select		train	publishr	dataa	leade				
PROcc51736f1f...	PROHIBITIONS	For the purpose...	TrainingMeasur...	PublishResults	DataAnalyst	Leaderboard	PROb235a3d1b...			
PER3fbf3e079b...	PERMISSIONS	For the purpose...	TrainingMeasur...	PublishResults	DataAnalyst	Leaderboard	PERab7189533...			

(1 of 1) 1 10

Figure 29: Found rules for given intention



General

Rule ID: PROcc51736f1f91490a8d23efbd7f3c6aac

Select Rule Type: PROHIBITIONS

Description: For the purpose of requested service provision, data analysts are forbidden to publish UserID information.

Variable Name: Purpose

Select Purpose: Leaderboard

Derived From Rule: PROb235a3d1bb9849c9919e445d6ee1eb28

Define Action

Variable Name: Action_Inferred

Select Resource: TrainingMeasurements

Select Operation: PublishResults

Select Actor: DataAnalyst

PreActions

PostActions

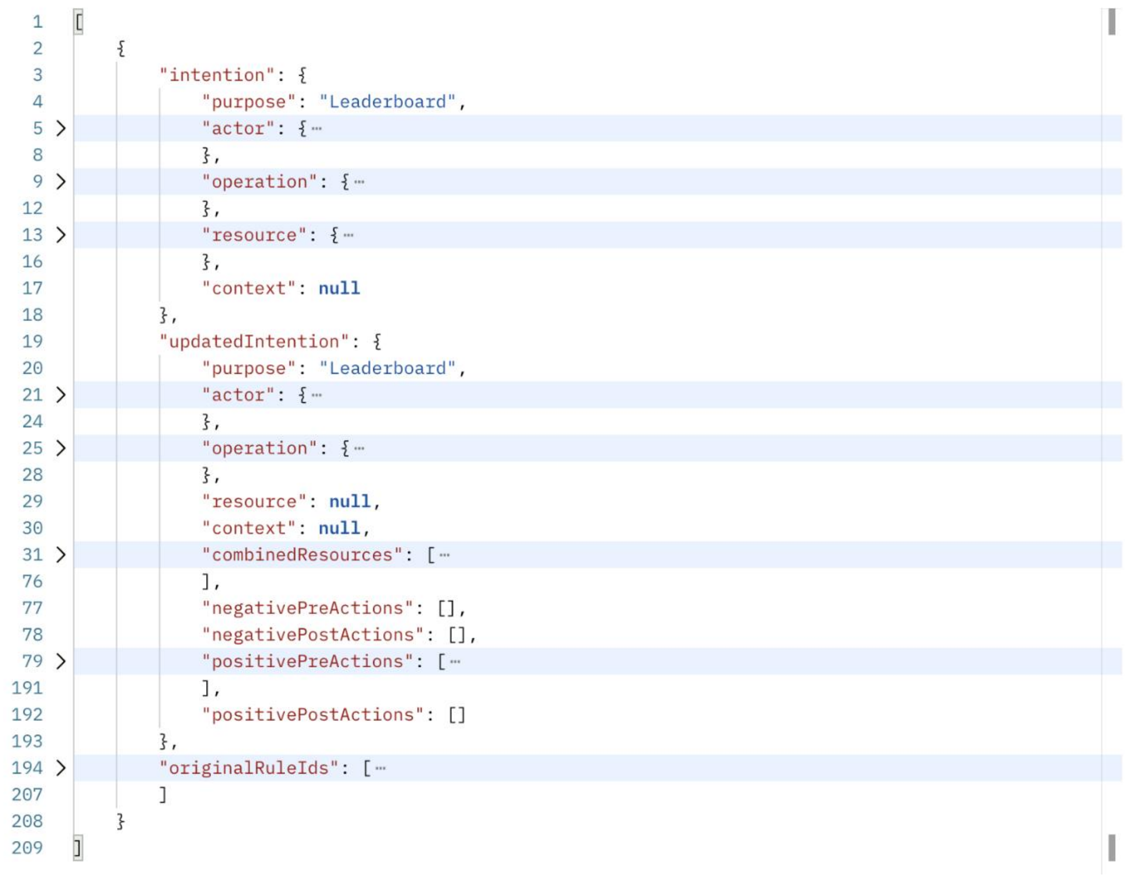
Define Contextual Condition

Figure 30: Inferred prohibition applying for the given intention

The response depicted in Figure 31 depicts the information made available by the conflict identification and resolution functionality. Specifically, the conflict structure contains the original intention, reference to the rules that lead to the result, and the updated intention, which apart from possibly changed properties of the given intention also contains ~~also~~:

- the combinedResources property, grouping together alternative resources to be accessed/used that are contained in originally requested resource
- positive/negative pre- and post- actions that should/not be performed.

In the case of our example, permissions to publish all but the UserID data types contained in the `TrainingMeasurements` data type are found, so the conflict resolution will suggest projection of the requested training measurements in order for `userID` to not be included in the published results, along with the execution of the required preActions of Figure 15 and Figure 16.



```
[
    {
        "intention": {
            "purpose": "Leaderboard",
            "actor": {⋯
            },
            "operation": {⋯
            },
            "resource": {⋯
            },
            "context": null
        },
        "updatedIntention": {
            "purpose": "Leaderboard",
            "actor": {⋯
            },
            "operation": {⋯
            },
            "resource": null,
            "context": null,
            "combinedResources": [⋯
            ],
            "negativePreActions": [],
            "negativePostActions": [],
            "positivePreActions": [⋯
            ],
            "positivePostActions": []
        },
        "originalRuleIds": [⋯
        ]
    }
]
```

Figure 31: Machine-readable representation of a conflict

In this version of the Policy Decision Point, these standalone intentions are evaluated against the Data Consumer's ruleset, while the intentions themselves are modelled leveraging constructs used in goodFlows operation, thus serving the validation of a Resource Consumer DPW step against the internal organisational policies. The next version of the API will include integration with the Resource Specification plugin; specifically, the privacy and usage constraints specified by the Resource Provider in ODRL will be translated and imported to a dedicated consumer-side policy model ontology, so as (i) intentions to be evaluated against both rulesets with the appropriate conflict resolution among rules and rule combination mechanisms in place, and (ii) updated intentions to be expressed in ODRL. To this end, the related ODRL adaptor is currently under construction.

Data processing workflow specification

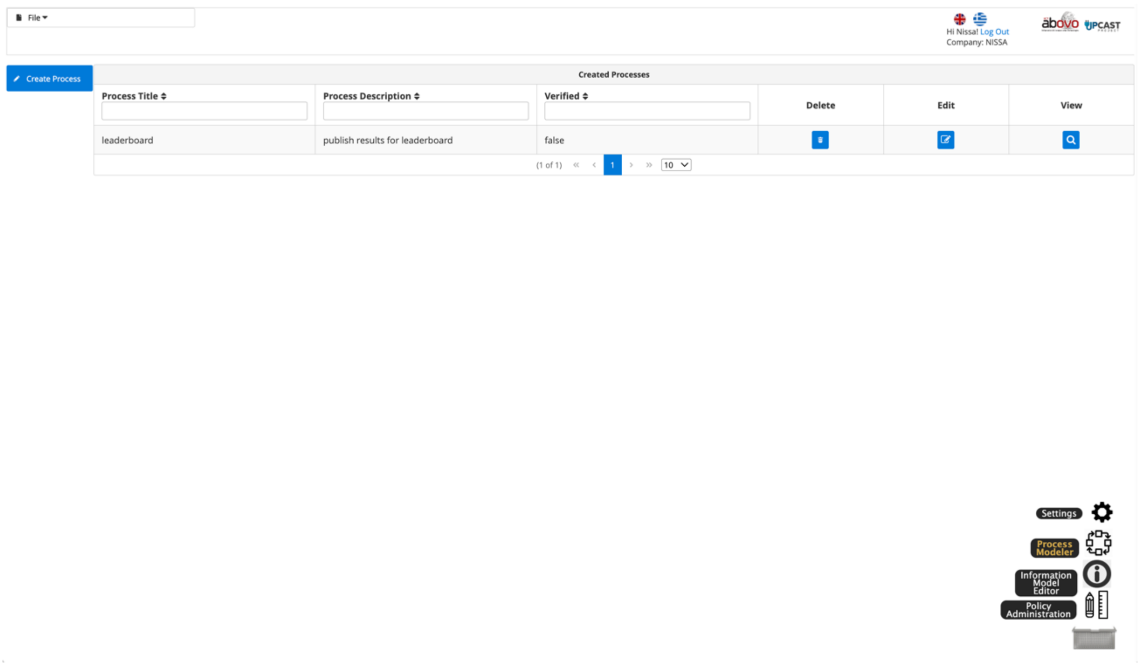
For defining UPCAST data processing workflows (DPWs), the starting point has been the Planning Environment offered by goodFlows; to this, adaptations and extensions are in the process of being introduced in order to accommodate, on the one hand, the more prevalent data orientation of UPCAST workflows, and, on the other, their necessary integration to other UPCAST plugins.

Of all the features to be offered by the planning environment, this deliverable focuses on its workflow modelling functionalities; these are made available through an appropriate REST API, devised for interacting with the Planning environment in a loosely-coupled fashion, and providing all appropriate functions for the creation and management of DPW models. The latter are expressed in goodFlows as ontologies; in that respect, the API covers all necessary functionality for the specification of processing tasks and flows, including actors, operations, assets, flow of information and all other aspects of a workflow, as well as its corresponding ontological representation. The current OpenAPI specification is shown at a high level in Annex 2.

The Planning environment is integrated, at a minimum, with the Information Model administration editor, as all processing tasks and flows are defined upon its entities. For the purposes of the current demonstrator, the goodFlows web-based user interface comprising both of these environments has been parameterised to the above REST API and to the selected use case (inspired by the UPCAST health and fitness pilot/ Nissatech). The use of the UI hides the technical details of the workflow model, requiring no particular technical expertise by its users; it translates the input provided through the graphical interface to machine code and the respective ontologies. Although additional features are expected to be added, an initial set of steps through which the end user will interact with the API through the envisioned front end are already established and summarised in the following:

1. View and manage data processing workflows

The introductory screen to the DPW plugin (Figure 32) calls, upon load, a method which returns all currently available DPW models, i.e., models that have been created and not in the meantime deleted, within the considered organisation or organisational unit. The table listing the DPW models offers some basic filtering functionality against DPW identifier, title, description text and verification status (to be expanded in D3.1 and D3.2), while, for each of the DPW models presented therein, the end-user is allowed to open it (view-only or edit mode, depending on authorisation level), or delete it. Finally, the user may choose to create a new DPW from scratch by clicking on “Create Process” button.



Created Processes

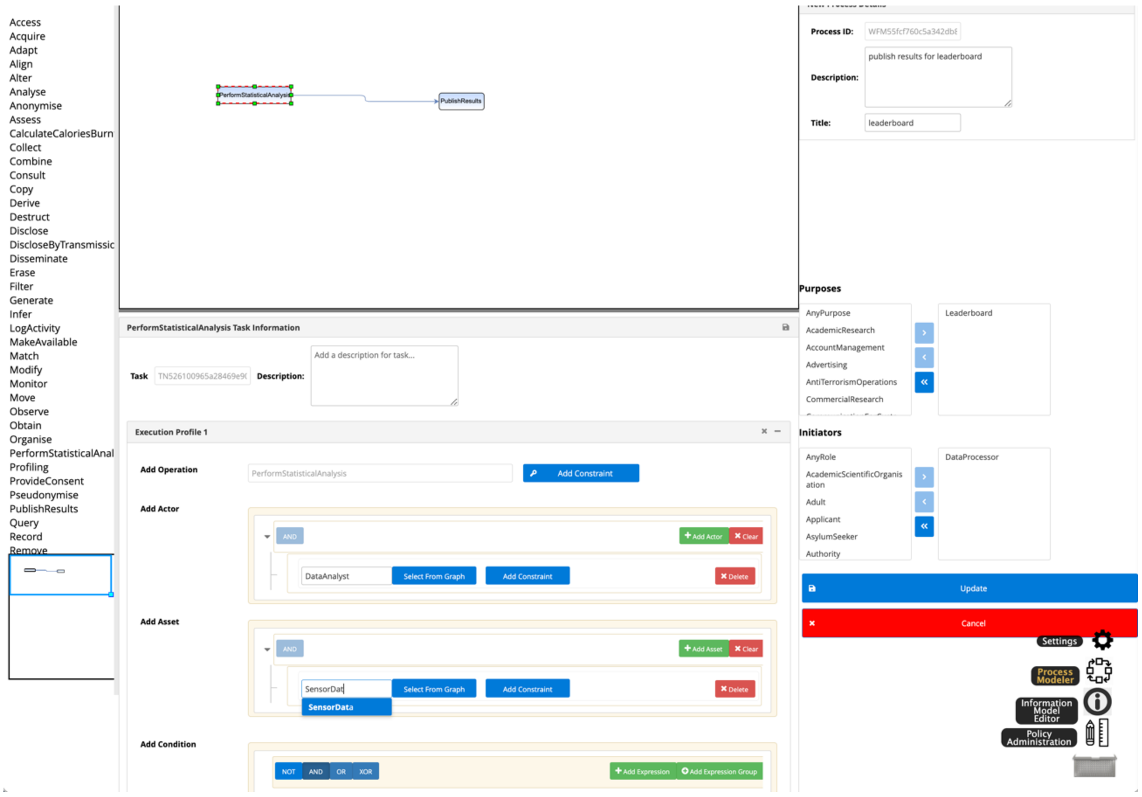
Process Title	Process Description	Verified	Delete	Edit	View
leaderboard	publish results for leaderboard	false			

(1 of 1) 1 10

Figure 32: Data Processing Workflow Management

2. Create a data processing workflow

By choosing to create a new DPW, the user lands on the main screen of the DPW plugin (Figure 33). The design area at the centre allows to graphically model a DPW by defining its tasks and flows. To the left, the environment presents the list of available processing operations defined for the particular organisation within its Information Model as part of the configuration of the tool, sorted alphabetically for facilitating navigation. To the right of the design area, the user is able to define a number of properties for the DPW model, including its association with purpose(s) intended to be served by it and the identification of the roles that may act as the initiators of the DPW. Finally, the UI is complemented by a map for making the navigation in the model easier.



Process ID: WFM55fcf760c5a342db
Description: publish results for leaderboard
Title: leaderboard

Purposes
AnyPurpose
AcademicResearch
AccountManagement
Advertising
AntiTerrorismOperations
CommercialResearch
Leaderboard

Initiators
AnyRole
AcademicScientificOrganisation
Adult
Applicant
AsylumSeeker
Authority
DataProcessor

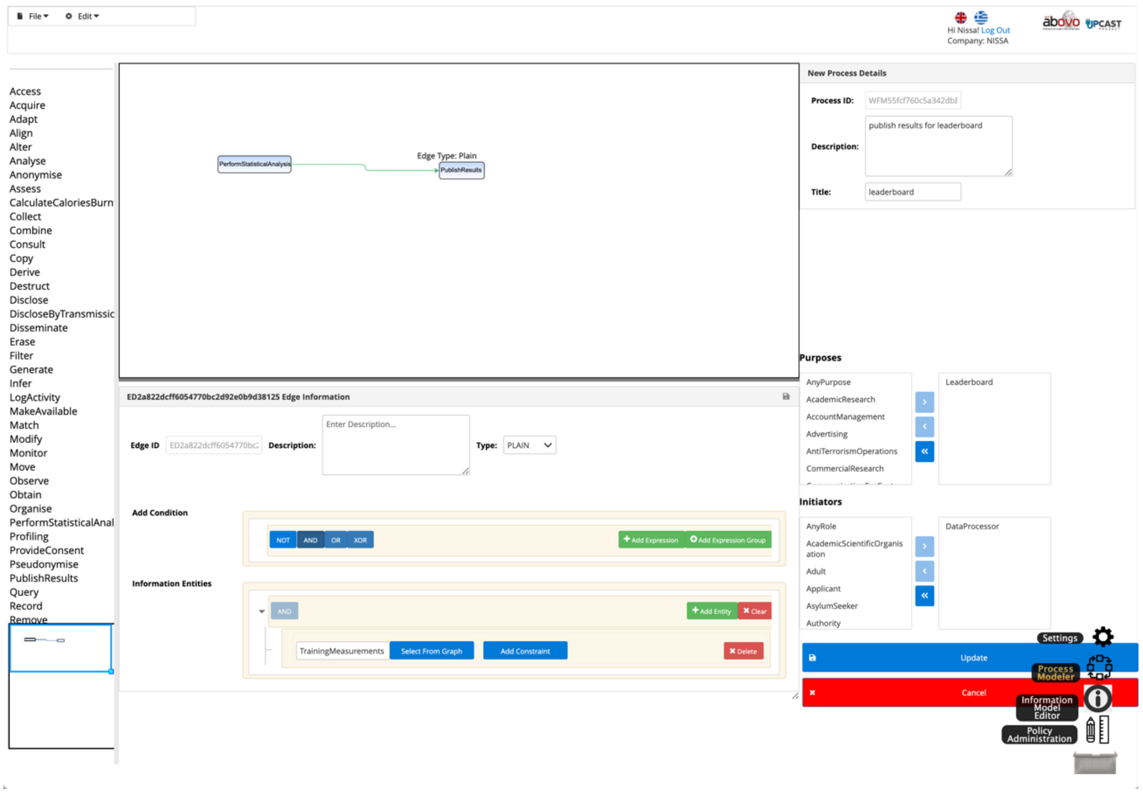
PerformStatisticalAnalysis Task Information
Task: TN526100965a28469e9 Description: Add a description for task...

Execution Profile 1
Add Operation: PerformStatisticalAnalysis
Add Actor: AND — DataAnalyst
Add Asset: AND — SensorDat | SensorData
Add Condition

Figure 33: DPW creation environment – Task PerformStatisticalAnalysis with its execution profile

The user defines purposes and initiators by selecting and de-selecting items from the respective lists, with the help of “plus” and “minus” buttons. As regards the population of the lists, their items are also here retrieved from the corresponding classes of the Information Model. In the example above the selected purpose is `Leaderboard`, whereas the selected initiator is `DataProcessor`, referring in a generic way to NISSA in DPV terms .

New tasks are created by dragging and dropping operations from the list on the left. For connecting tasks, i.e., defining the control and data flows among them, the user simply wires the boxes representing the tasks in the design area. This leads to the creation of edges, still neutral at this stage as regards their type (i.e., control or data flows). By clicking on an edge, however, the user has the option, through the dedicated area that dynamically appears below, to further define the kind of data the edge shall carry, as well as the flow type, or possible conditions that may apply; edge types, in particular, can also be updated directly by right-clicking on the edge. As an example, Figure 34 shows how the user may set the information entity carried by the edge connecting two tasks; here, this is defined to be of data type `TrainingMeasurements`, which is the output of the operation `PerformStatisticalAnalysis` performed by the edge source task. If required, additional constraints on the properties of selected information entities may be added through expressions and potential logical groups thereof. Further, by clicking on the “Add Entity” button, additional information entities may be associated with the same edge, depending on modelling requirements.



New Process Details

Process ID: WFM55fcf760c5a342dbE

Description: publish results for leaderboard

Title: leaderboard

Purposes: Leaderboard

Initiators: DataProcessor

ED2a822dcff6054770bc2d92e0b9d38125 Edge Information

Edge ID: ED2a822dcff6054770bc2

Description: Enter Description...

Type: PLAIN

Add Condition

Information Entities: TrainingMeasurements

Figure 34: DPW creation environment - Edge specification

Tasks in goodFlows can be seen as the cumulative effect of certain actors performing operations on well-defined assets. In order to allow for more flexibility in the definition of workflows, the tool has introduced the concept of *Execution Profiles* for defining distinct *<actor, operation, asset>* combinations as needed, which may therefore be used to denote alternative modes of executing the same task. By clicking on a task and then the “+” button below the design area, users may define one or more execution profiles bound to it, grouping together execution parameters, as well as actors and assets described in terms of their types and potential additional constraints. In the example of Figure 33 above, the execution profile added for task `PerformStatisticalAnalysis` defines as actor any individual holding the role `DataAnalyst`, while the asset upon which the task will be performed is of type `SensorData`. More than one actors and assets can be jointly defined in the same execution profile, subject to modelling requirements; for instance, instead of `SensorData`, the user might set the asset to specifically consist of `AccelerometerData` *and* `ElectrocardiographyData`, in order to denote that both subtypes of `SensorData`, but only them, should be used for the particular task according to the particular profile. Finally, conditions may also need to be added on occasion in order to further restrict the execution of a task as per the execution profile in question. The execution profile(s) of a task, together with the purpose(s) and initiator(s) defined for the DPW as a whole, can be seen as the overall *intention* of the DPW designer with respect to implied processing.

In defining tasks and edges in detail, information coming from the Information Model Editor, like e.g., data types, roles and attributes, become available either through pop-up dialogs (by pressing the “Select From Graph” button) or by typing in the respective textboxes with suggestions made by the system through autocompletion.

Finally, both tasks and edges can be deleted by right-clicking on them and explicitly proceeding with the deletion; in the case of deleted tasks, all associated edges are also removed from the DPW model.

3. Save a data processing workflow

Once they have created or modified a DPW model, the user may choose to save the result through the corresponding button at the bottom right of the main screen, named after “Save” or “Update” respectively. In any case, the DPW is saved in the underlying RDF Triplestore, while the graph representation of the DPW generated by the UI is also stored as an mxgraph zip file for future use.

4. Open a data processing workflow

Upon opening an existing DPW model, its graph representation is handed over to the UI in the form of an mxgraph zip file. Tasks, edges and all other DPW elements are retrieved and displayed to the user through the same screen as shown in Figure 24 and Figure 25. In case a DPW has been opened for editing, the exact same functionalities described in the creation step above are offered; if a DPW is viewed without editing rights, all clickable and modification functionality is disabled/not rendered.

5. Export a data processing workflow

The user is enabled to export the DPW model as an OWL ontology and retrieve it as an OWL file. In its enriched version this can also contain the full Information Model and the basic concepts of Workflow Models.

3 Federated Machine Learning Plugin

Federated Machine Learning (FML) is a machine learning technique that enables training models on distributed datasets without the need to centralize the data. In FML, the model is trained across multiple decentralized edge devices or servers holding local data samples, without exchanging them. This approach addresses critical issues around data privacy, security, access rights, and heterogeneous data.

There are multiple benefits of FML:

- **Data Privacy**: Raw data never leaves the local devices, protecting user privacy.
- **Regulatory Compliance**: Helps in complying with data protection regulations like GDPR.
- **Reduced Data Transfer**: Only model updates are shared, reducing bandwidth usage.
- **Access to Diverse Data**: Enables learning from a wide range of data sources without centralization.
- **Continuous Learning**: Models can be updated frequently with fresh, real-world data.
- **Scalability**: Can handle large numbers of participants and huge amounts of decentralized data.
- **Reduced Central Storage Need**: No need for a large central data repository.
- **Collaboration**: Allows multiple organizations to collaborate without sharing sensitive data.

A usage scenario in the context of UPCAST is as follows, a Data Consumer needs to train a Machine Learning model. She discovers in a Data Marketplace multiple Data Providers that own data that could be useful to train the model. However, the usage constraints of the Data Providers prohibit the transfer of data outside of their premises. Providers are still interested in doing business with Data Consumer and are willing to use their own computational resources to collaborate with the Consumer without having to transfer data.

We regard FML as a special kind of Data Operation, we assume the FML plugin is available in the resource catalog of the Data Marketplace used (or IDSA DataApp Store). Its metadata can be then used as a step in a Data Processing Workflow, connecting the FML plugin with the rest of the UPCAST infrastructure. The continuation of the usage scenario above is as follows:

1. The Data Consumer creates a Data Processing Workflow including a FML step. For ease of exposition, we assume FML is the only step.
2. The Data Consumer negotiates with each provider the usage conditions of each

dataset, assisted by the DPW plugin to check there are no conflicts between the agreements of multiple providers.

3. Once all negotiations have been successfully completed, the FML plugin provides the Data Consumer and Data Providers with the setup need to train a Machine Learning model in a Federated way.

Figure 35 describes the general setup of the FML plugin. The central server coordinates the learning process and aggregates model updates. Clients (edge devices or local servers) train the model on their local data. Only model updates are shared, not the raw data.

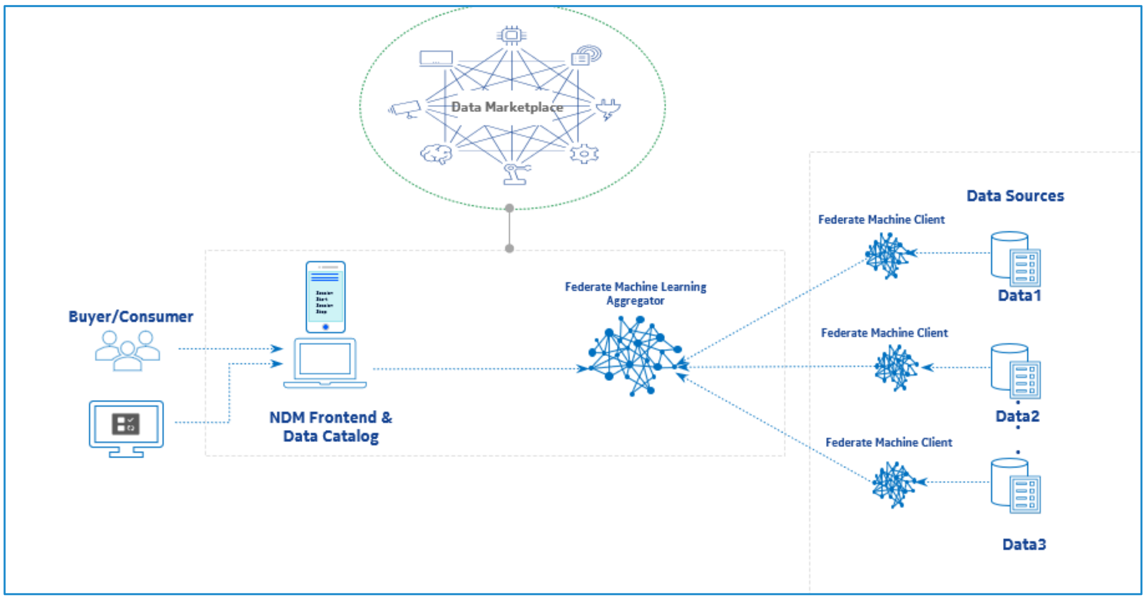


Figure 35: Federated Machine Learning plugin high level design. Nokia Data Marketplace (NDM) used as example Data Marketplace

An important condition that needs to be agreed during negotiation is that Data Providers must make available computing infrastructure to participate in the Federated training of a ML model. They must be able to install the FML plugin agent on their server and their server should be able to run a machine learning process on the local data.

3.1 Plugin Structure

The project consists of two main components: the server and the client.

3.1.1 Server

Server app which should be run on the main server side or inside a marketplace is acting as an orchestrator and aggregator in a FML project. The API of the Server component is available on the UPCAST github repository[2]

The main tasks of "server" are:

- Flask application serving as the central aggregator.
- Manages FML sessions and model aggregation.
- Provides API endpoints for setup, model upload, and training initiation.
- Uses SQLite database to store session configurations.
- Implements the `Strategy` for model aggregation.

3.1.2 Client

The client app, which acts as an agent on the client side, manages the local ML process and communicates the model parameters to the server side.

The main tasks of the client side:

- Simulates edge devices or local data holders
- Implements local model training
- Communicates with the server to receive and send model updates

3.2 Requirements, setup and run instructions

As it has been mentioned before, in the FML scenarion there will be a constant communication between the clients(data providers) and the server. For this purpose clients should provide an server(or VM) with open port and public IP so the server would be able to send the model parameters and start the FML process

To run an FML session, the following steps must be followed.

Setup Environment:

- Install required dependencies (Flask, Flower, PyTorch, etc.)
- Ensure Python 3.7+ is installed

Upload Initial Model:

- Use the `/uploadmodel` endpoint to upload an initial PyTorch model

[2] https://github.com/EU-UPCAST/OpenAPISpecification/tree/main/fml_plugin

Create FML Session:

- Use the `/savesetup` endpoint to create a new FML session
- Specify parameters like strategy, number of rounds, and minimum clients

Initiate Training:

- Use the `/start` endpoint to begin the federated learning process
- Provide the initial model path, number of clients, and rounds

Start Clients:

- Run multiple instances of `client.py` on different machines or processes
- Ensure clients are configured with the correct server address

Monitor Progress:

- Observe server logs for round completions and model updates
- Use WebSocket connections to receive real-time updates

Retrieve Final Model:

- After all rounds are complete, the final aggregated model is saved as "final_global_model.pth"

Analyze Results:

- Evaluate the final model's performance
- Compare with initial model to assess improvement

3.3 FML process

When the Training is Initiated, these steps are automated:

1. Server Initialization:

The central server prepares for the federated learning process. It loads the initial model and sets up the SaveModelStrategy with the specified parameters (number of rounds, number of clients, etc.).

2. Client Selection:

The server selects a subset of available clients to participate in the current round of training. In your implementation, all available clients are selected for each round.

3. Model Distribution:

The server sends the current global model parameters to all selected clients. This is done without transferring the entire model structure, just the weights and biases.

4. Local Training:

Each client receives the global model parameters and updates them with their local data:

- The client loads the received parameters into their local model.
- It trains the model on its local dataset for a specified number of epochs.
- During this process, the model learns patterns specific to the client's local data.

5. Model Update Calculation:

After local training, each client calculates the difference between the new model parameters and the original ones received from the server. This difference represents the "model update" or "gradient" for that client.

6. Sending Updates to Server:

Clients send only these calculated updates back to the server, not their raw data or the entire model. This step is crucial for maintaining data privacy.

7. Update Aggregation:

The server receives updates from all participating clients and aggregates them. In your implementation, this is done using the FedAvg (Federated Averaging) algorithm, which essentially takes a weighted average of all client updates.

8. Global Model Update:

The server applies the aggregated update to the global model, creating a new version that incorporates learning from all participating clients.

9. Model Evaluation:

Optionally, the server may evaluate the updated global model on a test dataset to track its performance.

10. Progress Tracking:

The server logs the training progress, including metrics like the update norm, which indicates how much the model has changed in this round.

11. WebSocket Update:

Real-time updates about the training progress are sent to connected clients (e.g., a web interface) via WebSocket, allowing for live monitoring of the process.

12. Round Completion:

The current round is marked as complete. If there are more rounds to go, the process repeats from step 2.

13. Final Model Saving:

After all rounds are complete, the final global model is saved to disk.

Throughout this process, several key aspects of federated learning are maintained:

- **Privacy**: Raw data never leaves the clients; only model updates are shared.
- **Efficiency**: The server aggregates learning from multiple sources without needing access to the raw data.
- **Collaboration**: Multiple clients contribute to improving a shared model.
- **Adaptability**: The global model learns from diverse data sources, potentially improving its generalization.

This cycle of local computation and global aggregation allows for the creation of a powerful, collaboratively trained model while preserving the privacy and security of each participant's data.

4 Conclusions and Next Steps

This document describes the version 1 of UPCAST plugins Privacy and Usage Control, and Federated Machine Learning.

Producer side of Privacy Plugin:

We will work on the definition of complementary actions, that is, providing further fine-grained control to Producers with regards to data, including the option to define rules that are applicable to a subset of a dataset. We will also Incorporate a policy engine on the Producer side to enable decisions of access and usage of data.

Consumer side of Privacy Plugin:

It currently offers specification of consumer polices, reasoning over them for policy decision including conflict resolution, as well as specification of consumer DPWs. In this version of the policy decision mechanism, standalone intentions are evaluated against the Data Consumer's ruleset, thus serving the validation of a Resource Consumer DPW step against the internal organisational policies. The next version of the API will include integration with the Resource Specification plugin; specifically, the privacy and usage constraints specified by the Resource Provider in ODRL will be translated and imported to a dedicated consumer-side policy model ontology, so as (i) intentions to be evaluated against both rulesets with the appropriate conflict resolution among rules and rule combination mechanisms in place, and (ii) updated intentions to be expressed in ODRL. To this end, the related ODRL adaptor is currently under construction. Further, certain aspects of the DPW plugin are still in progress and will be documented in upcoming deliverables, mainly within WP3. These include: extension with and adaptations in support of additional UPCAST-specific requirements (e.g., involve datasets and data products as first-class citizens, provide direct linkage to the execution environment, and facilitate additional DPW-wide constraints addressing pricing or energy footprint considerations); introduction of new notation into the DPW design tool to support above enhancements (e.g., for datasets and FML operations); aspects pertaining to the interoperation with other data plugins, like Resource Specification & Discovery, Negotiation & Contracting, Pricing, Environmental impact optimiser, etc.

Federated Machine Learning:

Advance on the GUI of the plugin to improve usability, and of the packaging of the necessary software agents to facilitate installation.

ANNEX I ACRONYMS & ABBREVIATIONS

Acronyms

Table 1: Acronyms List

Acronyms List	
CP	Consortium Plenary
DoA	Description of Action
PC	Project Coordinator
PMB	Project Management Board
PPR	Project Periodic Report
QM	Quality Management
RM	Risk Management
TM	Technical Manager
WPL	Work Packages Leaders
DMP	Data Management Plan
GDPR	General Data Protection Regulation
RBS	Risk Breakdown Structure

Abbreviations

Table 2: Acronyms List

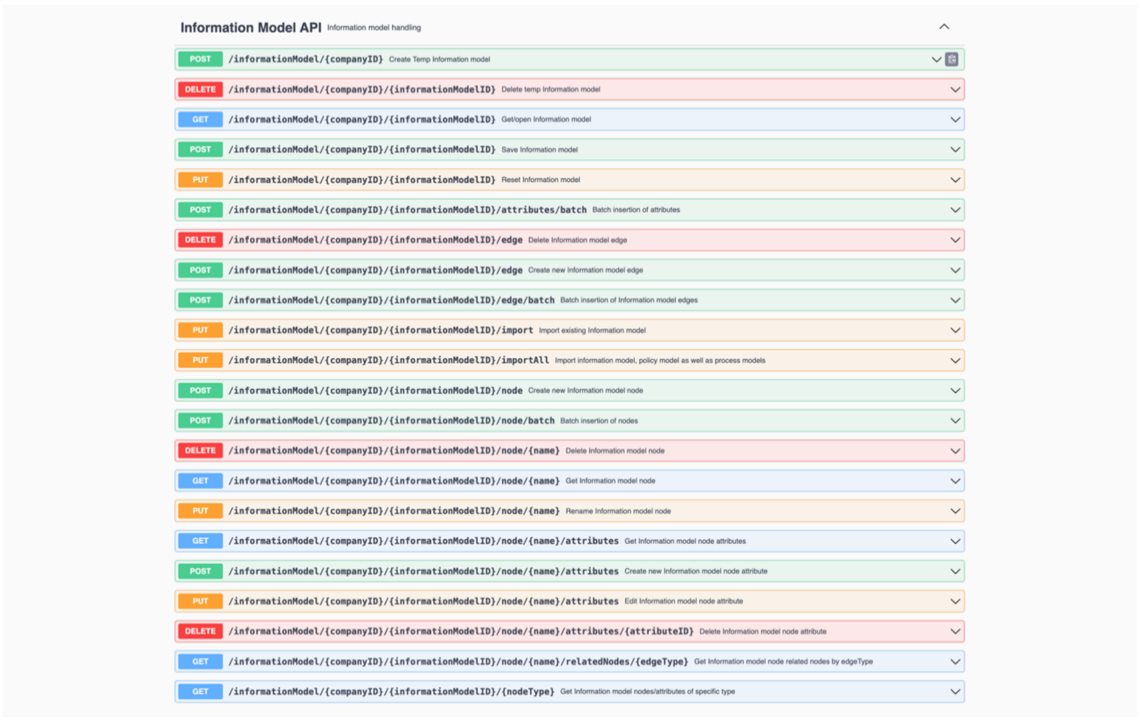
Abbreviation List

API	Application Programming Interface - A set of rules and protocols that allows different software applications to communicate and interact with each other.
DCAT	Data Catalog Vocabulary is a metadata standard used to describe datasets, data catalogs, and data portals.
DMP	A document that outlines how data will be handled throughout the research project, including data collection, storage, sharing, and preservation.
DOI	Digital Object Identifier - A unique alphanumeric identifier assigned to a digital object, such as a dataset or publication, to provide a persistent link to its location on the internet.
ETL	Extract, Transform, Load - The process of extracting data from various sources, transforming it into a consistent format, and loading it into a target system or database.
FAIR	Findable, Accessible, Interoperable, and Reusable - A set of principles that aim to make data discoverable, accessible, and usable by both humans and machines.
GDPR	The European Union regulation that governs the protection and privacy of personal data. It sets guidelines for data processing, consent, and individuals' rights.
NDA	Non-Disclosure Agreement - A legal contract that establishes confidentiality obligations between parties involved in sharing sensitive or proprietary information.
NFR	Non-Functional Requirements - Requirements that specify the characteristics and qualities of a system, such as security, performance, scalability, and usability.
PII	Personally Identifiable Information - Information that can be used to identify an individual, such as name, address, contact details, or unique identifiers.
Pseudonymization	The process of replacing identifiable data with artificial identifiers, called pseudonyms, to protect individual privacy while still allowing data analysis and processing.
TLS	Transport Layer Security is one of the most common encryption protocols used to secure data during transmission over a network.

ANNEX II Open API Specification

Available on the Github Repository.

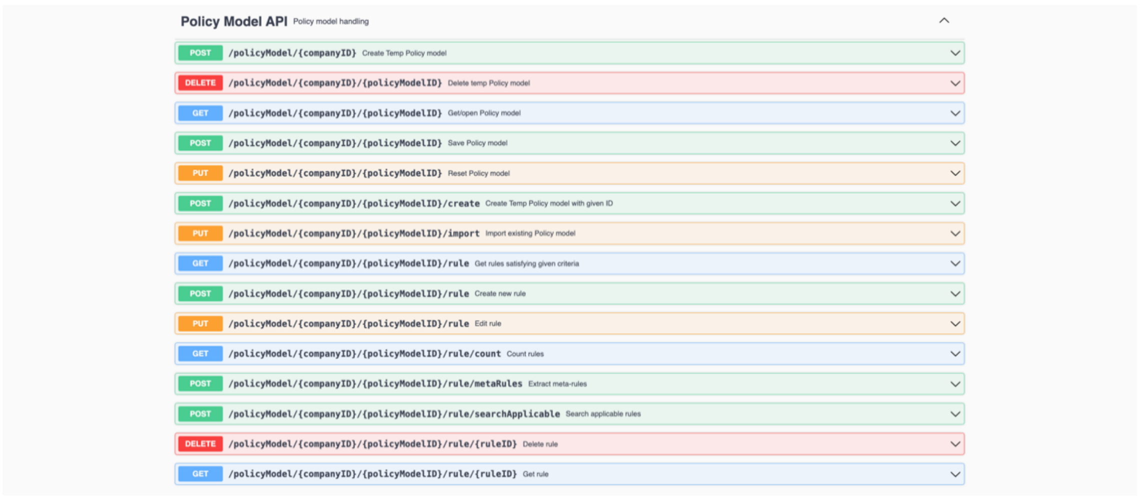
Privacy plugin: https://github.com/EU-UPCAST/OpenAPISpecification/tree/main/privacy_plugin



Information Model API Information model handling

```
POST    /informationModel/{companyID} Create Temp Information model
DELETE  /informationModel/{companyID}/{informationModelID} Delete temp Information model
GET     /informationModel/{companyID}/{informationModelID} Get/open Information model
POST    /informationModel/{companyID}/{informationModelID} Save Information model
PUT     /informationModel/{companyID}/{informationModelID} Reset Information model
POST    /informationModel/{companyID}/{informationModelID}/attributes/batch Batch insertion of attributes
DELETE  /informationModel/{companyID}/{informationModelID}/edge Delete Information model edge
POST    /informationModel/{companyID}/{informationModelID}/edge Create new Information model edge
POST    /informationModel/{companyID}/{informationModelID}/edge/batch Batch insertion of Information model edges
PUT     /informationModel/{companyID}/{informationModelID}/import Import existing Information model
PUT     /informationModel/{companyID}/{informationModelID}/importAll Import information model, policy model as well as process models
POST    /informationModel/{companyID}/{informationModelID}/node Create new Information model node
POST    /informationModel/{companyID}/{informationModelID}/node/batch Batch insertion of nodes
DELETE  /informationModel/{companyID}/{informationModelID}/node/{name} Delete Information model node
GET     /informationModel/{companyID}/{informationModelID}/node/{name} Get Information model node
PUT     /informationModel/{companyID}/{informationModelID}/node/{name} Rename Information model node
GET     /informationModel/{companyID}/{informationModelID}/node/{name}/attributes Get Information model node attributes
POST    /informationModel/{companyID}/{informationModelID}/node/{name}/attributes Create new Information model node attribute
PUT     /informationModel/{companyID}/{informationModelID}/node/{name}/attributes Edit Information model node attribute
DELETE  /informationModel/{companyID}/{informationModelID}/node/{name}/attributes/{attributeID} Delete Information model node attribute
GET     /informationModel/{companyID}/{informationModelID}/node/{name}/relatedNodes/{edgeType} Get Information model node related nodes by edgeType
GET     /informationModel/{companyID}/{informationModelID}/{nodeType} Get Information model nodes/attributes of specific type
```

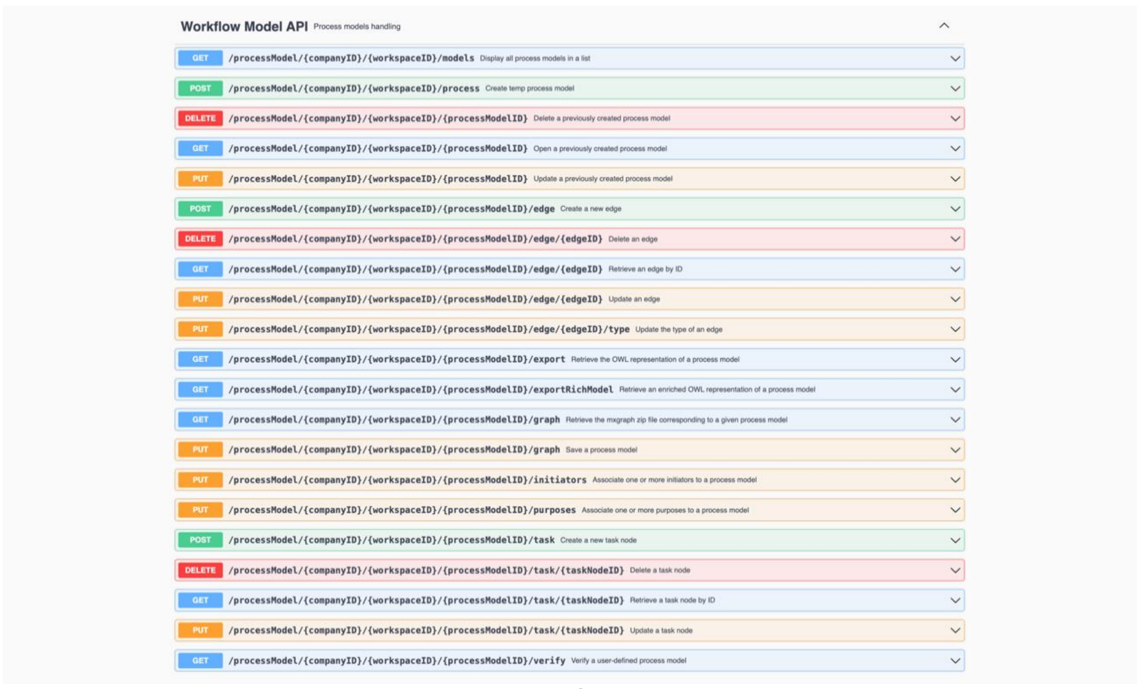
Figure 36: Information Model API



Policy Model API Policy model handling

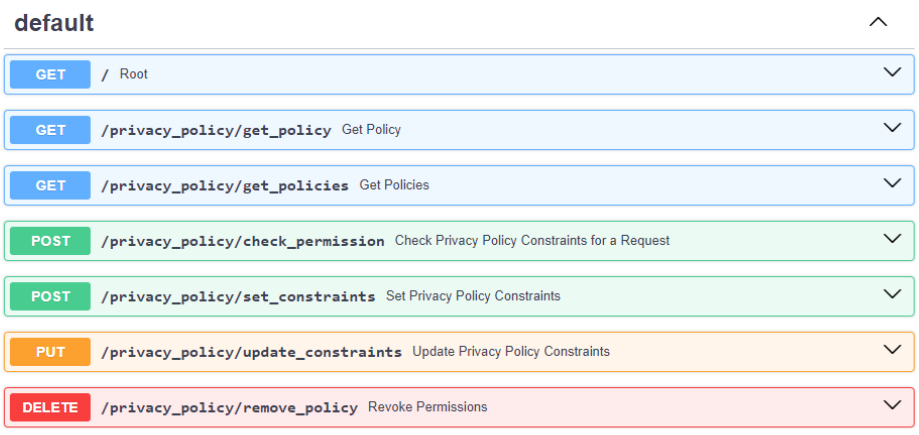
```
POST    /policyModel/{companyID} Create Temp Policy model
DELETE  /policyModel/{companyID}/{policyModelID} Delete temp Policy model
GET     /policyModel/{companyID}/{policyModelID} Get/open Policy model
POST    /policyModel/{companyID}/{policyModelID} Save Policy model
PUT     /policyModel/{companyID}/{policyModelID} Reset Policy model
POST    /policyModel/{companyID}/{policyModelID}/create Create Temp Policy model with given ID
PUT     /policyModel/{companyID}/{policyModelID}/import Import existing Policy model
GET     /policyModel/{companyID}/{policyModelID}/rule Get rules satisfying given criteria
POST    /policyModel/{companyID}/{policyModelID}/rule Create new rule
PUT     /policyModel/{companyID}/{policyModelID}/rule Edit rule
GET     /policyModel/{companyID}/{policyModelID}/rule/count Count rules
POST    /policyModel/{companyID}/{policyModelID}/rule/metaRules Extract meta-rules
POST    /policyModel/{companyID}/{policyModelID}/rule/searchApplicable Search applicable rules
DELETE  /policyModel/{companyID}/{policyModelID}/rule/{ruleID} Delete rule
GET     /policyModel/{companyID}/{policyModelID}/rule/{ruleID} Get rule
```

Figure 37: Policy Model API



```
Workflow Model API  Process models handling
GET     /processModel/{companyID}/{workspaceID}/models  Display all process models in a list
POST    /processModel/{companyID}/{workspaceID}/process  Create temp process model
DELETE  /processModel/{companyID}/{workspaceID}/{processModelID}  Delete a previously created process model
GET     /processModel/{companyID}/{workspaceID}/{processModelID}  Open a previously created process model
PUT     /processModel/{companyID}/{workspaceID}/{processModelID}  Update a previously created process model
POST    /processModel/{companyID}/{workspaceID}/{processModelID}/edge  Create a new edge
DELETE  /processModel/{companyID}/{workspaceID}/{processModelID}/edge/{edgeID}  Delete an edge
GET     /processModel/{companyID}/{workspaceID}/{processModelID}/edge/{edgeID}  Retrieve an edge by ID
PUT     /processModel/{companyID}/{workspaceID}/{processModelID}/edge/{edgeID}  Update an edge
PUT     /processModel/{companyID}/{workspaceID}/{processModelID}/edge/{edgeID}/type  Update the type of an edge
GET     /processModel/{companyID}/{workspaceID}/{processModelID}/export  Retrieve the OWL representation of a process model
GET     /processModel/{companyID}/{workspaceID}/{processModelID}/exportRichModel  Retrieve an enriched OWL representation of a process model
GET     /processModel/{companyID}/{workspaceID}/{processModelID}/graph  Retrieve the mxgraph zip file corresponding to a given process model
PUT     /processModel/{companyID}/{workspaceID}/{processModelID}/graph  Save a process model
PUT     /processModel/{companyID}/{workspaceID}/{processModelID}/initiators  Associate one or more initiators to a process model
PUT     /processModel/{companyID}/{workspaceID}/{processModelID}/purposes  Associate one or more purposes to a process model
POST    /processModel/{companyID}/{workspaceID}/{processModelID}/task  Create a new task node
DELETE  /processModel/{companyID}/{workspaceID}/{processModelID}/task/{taskNodeID}  Delete a task node
GET     /processModel/{companyID}/{workspaceID}/{processModelID}/task/{taskNodeID}  Retrieve a task node by ID
PUT     /processModel/{companyID}/{workspaceID}/{processModelID}/task/{taskNodeID}  Update a task node
GET     /processModel/{companyID}/{workspaceID}/{processModelID}/verify  Verify a user-defined process model
```

Figure 38: Workflow Model API



```
default
GET     /  Root
GET     /privacy_policy/get_policy  Get Policy
GET     /privacy_policy/get_policies  Get Policies
POST    /privacy_policy/check_permission  Check Privacy Policy Constraints for a Request
POST    /privacy_policy/set_constraints  Set Privacy Policy Constraints
PUT     /privacy_policy/update_constraints  Update Privacy Policy Constraints
DELETE  /privacy_policy/remove_policy  Revoke Permissions
```

Figure 39: PrivUI API