smashHit

User & Developer Guide

Data Provider

Data Processor

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 871477

Introduction

Purpose

This guide aims to help organization users and developers with a data “provider” and/or “processor” technical role[1] on how to use the smashHit platform and gives knowledge about the different functionalities available.

Audience

This guide is meant for and solely for organization users of the smashHit platform with data “provider” or “processor” role. Other roles can find their own user guides under smashhit.eu.

Scope

The contents of this guide are meant to be taken into consideration only when using the smashHit platform and will only cover functionalities meant to be used by the roles stated above.

The smashHit team does not take responsibility for improper use of the application or the data provided when not following the instructions given in this guide.

Troubleshooting

For any questions or inquiries about the use of the smashHit platform web application or the contents of it or this guide, or if you find there is no content in this guide for some functionality, please forward it to: info@smashhit.eu

Contact

smashHit Project website: https://smashhit.eu

smashHit platform support: info@smashhit.eu

[1] Organizations “providing” the data or just “processing” data (in the technical sense of the term) might adopt either a data “controller” or “processor” role as stated by GDPR in different scenarios, to be defined in a case-by-case basis, depending on the terms for the processing of personal data.

Contents

Introduction 1
Purpose 1
Audience 1
Scope 1
Troubleshooting 1
Contact 1
Contents 2
Guide 3
Basic knowledge 3
Step by step 3
Registration 3
Application management 6
Consent template management 9
Consent management 12
Notification endpoint 17
Data use traceability 17
Automatic contracting 21
Context-Sensitive Security and Privacy 24
F.A.Q. 27
smashHit platform 27
Glossary 29
Figures 30

Guide

Basic knowledge

The objective of the smashHit framework is to assure trusted and secure sharing of data streams from both personal and industrial platforms, needed to build sectorial and cross-sectorial services, by establishing a framework to facilitate the processing of data subject consent and legal rules and effective contracting, as well as joint security and privacy preserving mechanisms. While these data streams offer new opportunities to build innovative services, they often require the management of a complex set of consent chains.

The main goal of smashHit is thus to overcome these obstacles by offering a set of methods and tools to facilitate the management of consents, supported by semantic models of consent and legal rules. The new tools include traceability of use of data, data fingerprinting and automatic contracting among the data subjects, data controllers and data processors.

Step by step

Data provider/processor organizations in smashHit are meant to be managed thorough the usage of “organization user” accounts. These organization users can manage both administrative and developer-oriented tasks, ranging from the definition of consent templates through the web interface, to the integration with smashHit APIs for consent management. This guide will cover the different options and processes to follow in order for an organization to integrate their application(s) with the smashHit framework.

The following steps would represent the typical tasks to be performed by an organization user in order to make use of smashHit:

- Organization registration request
- First login and additional organization users’ setup
- Application registration and configuration
- Consent template definition
- Consent creation and management
- Consent notifications
- Contract management
- Use of Traceability component.

Registration

The first step for organization users would be to fill a registration form. From the landing page[2], users can click the “Sign Up!” button to submit a registration request in smashHit (Figure 1) that will be reviewed and accepted (or rejected) by an administrator.

[2] https://smashhit.ari-mobility.eu

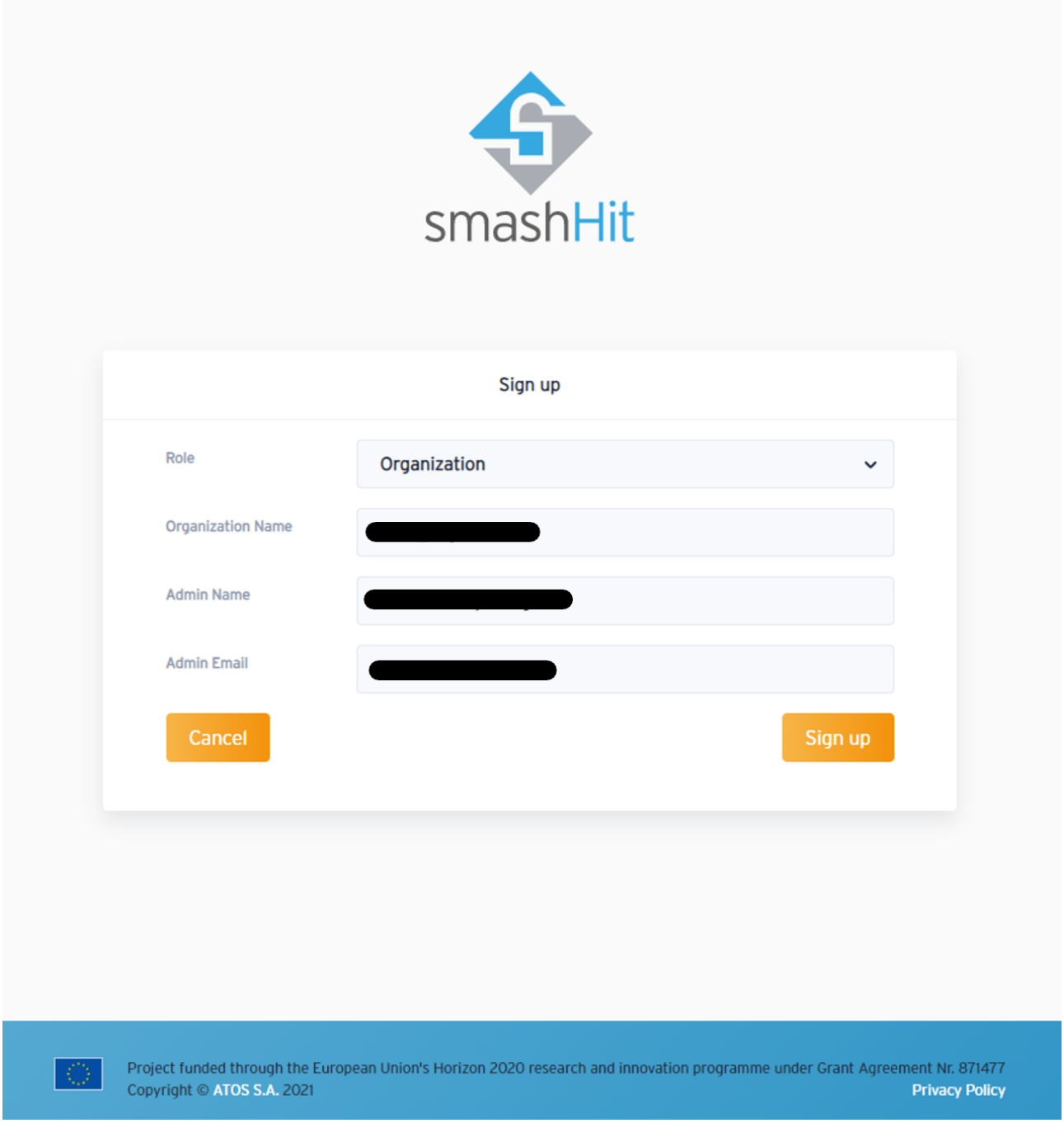


Figure 1: Organization registration

As the administrator validates the organization registration request, an email will be sent to the address specified by the organization with instructions on how to access smashHit. Now, from the landing page, organization users can click “Sign in” and log in with their assigned credentials, as seen in Figure 2.

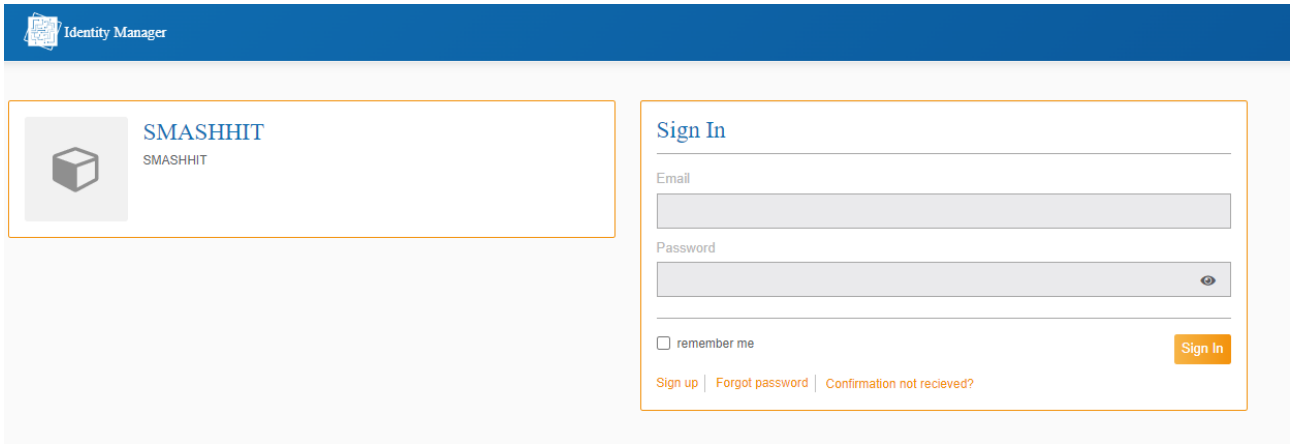


Figure 2: Sign in

Once signed in in smashHit, organization users will be shown a quick resume of their organization details & organization admin users, as seen in Figure 3.

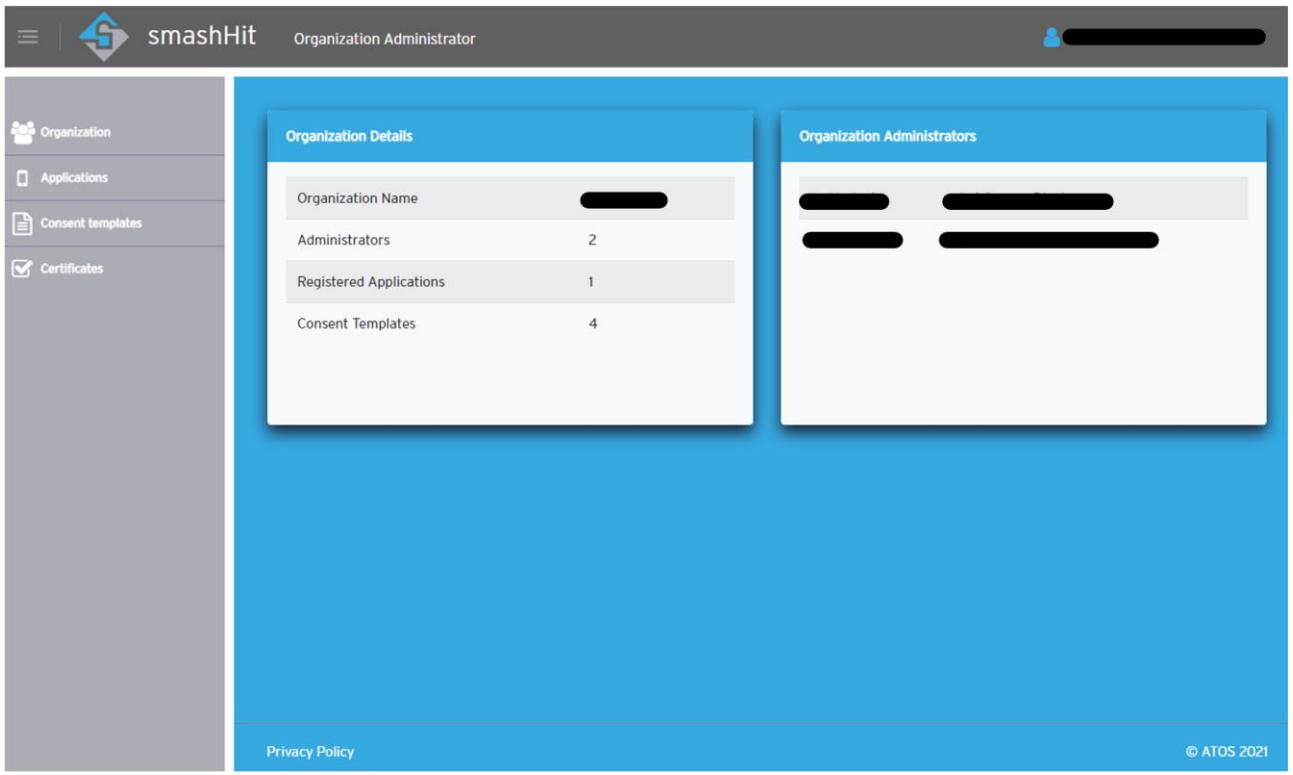


Figure 3: Organization landing page

In the left sidebar, the organization user can click “Organization”, “Application”, “Consent templates”, or “Certificates” to access the different functionalities.

Clicking “Organization”, will open the interface seen in Figure 4, where the user can see a brief list of their applications and consent templates items, as well as add or remove additional users with admin rights for the organization.

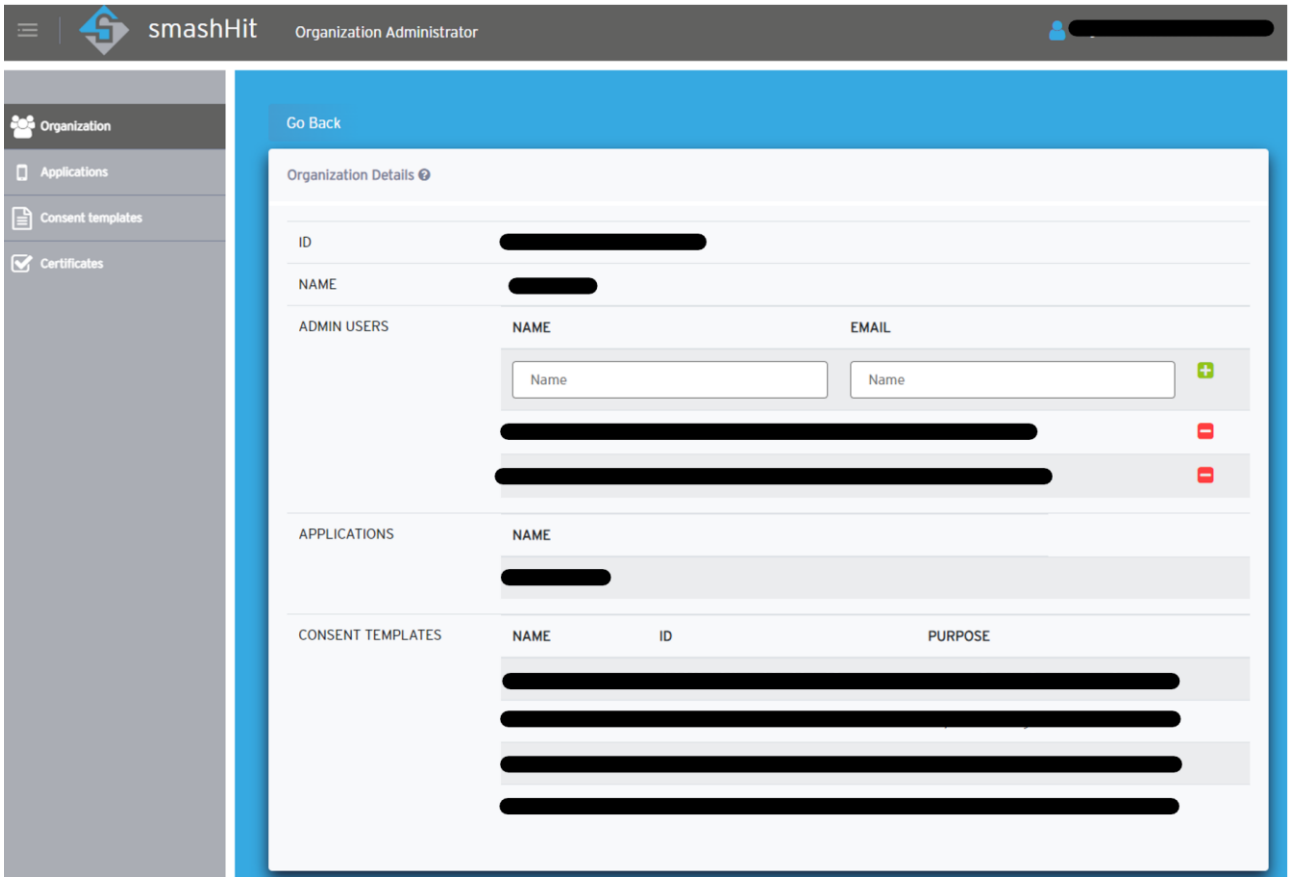


Figure 4: Organization management

Application management

As an organization, the first step to integrate an application that will interact with users' personal data is to model the application itself in the platform through the "Application" tab.

In the context of smashHit, an application represents specific pieces of software owned by an organization, that will process personal data, and thus has the legal requirement to request for the data subject's consent. The concept of application is an abstraction for organizations to easily aggregate a set of consents used by the same application.

Depending on the type of processing to be done by the application, the organization might take a data controller or processor role, something which will be defined later in the consent template for each case.

From the "Applications" tab (Figure 5), the organization can manage existing applications for their organization (Figure 6) or register a new one (Figure 7).

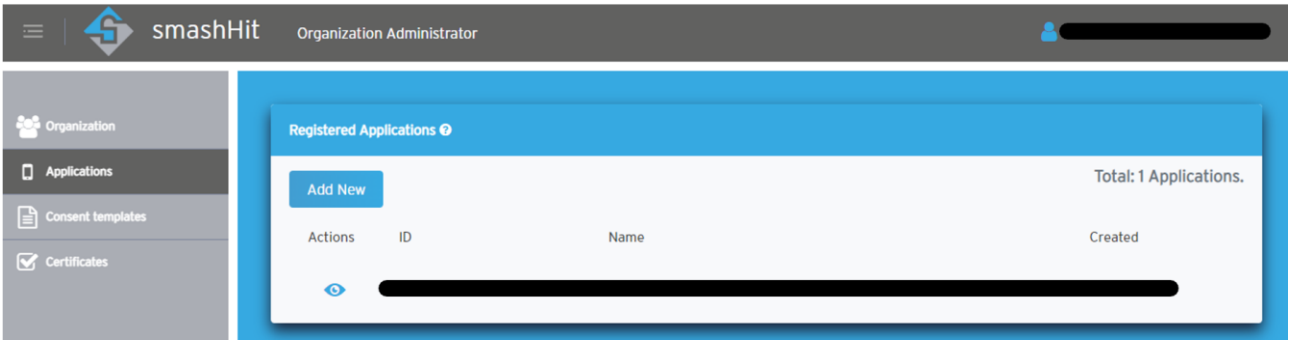


Figure 5: Applications list

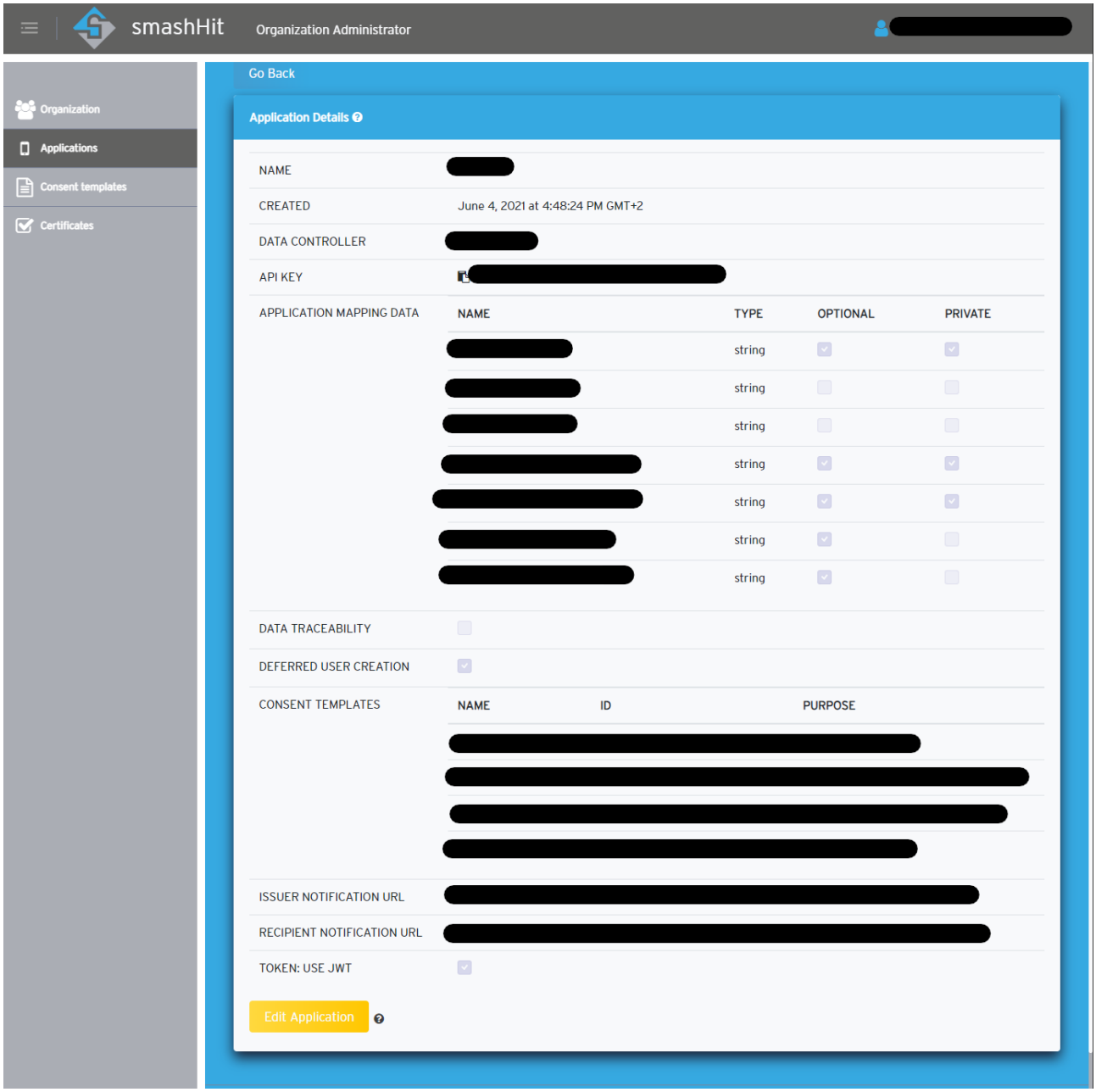


Figure 6: Application management

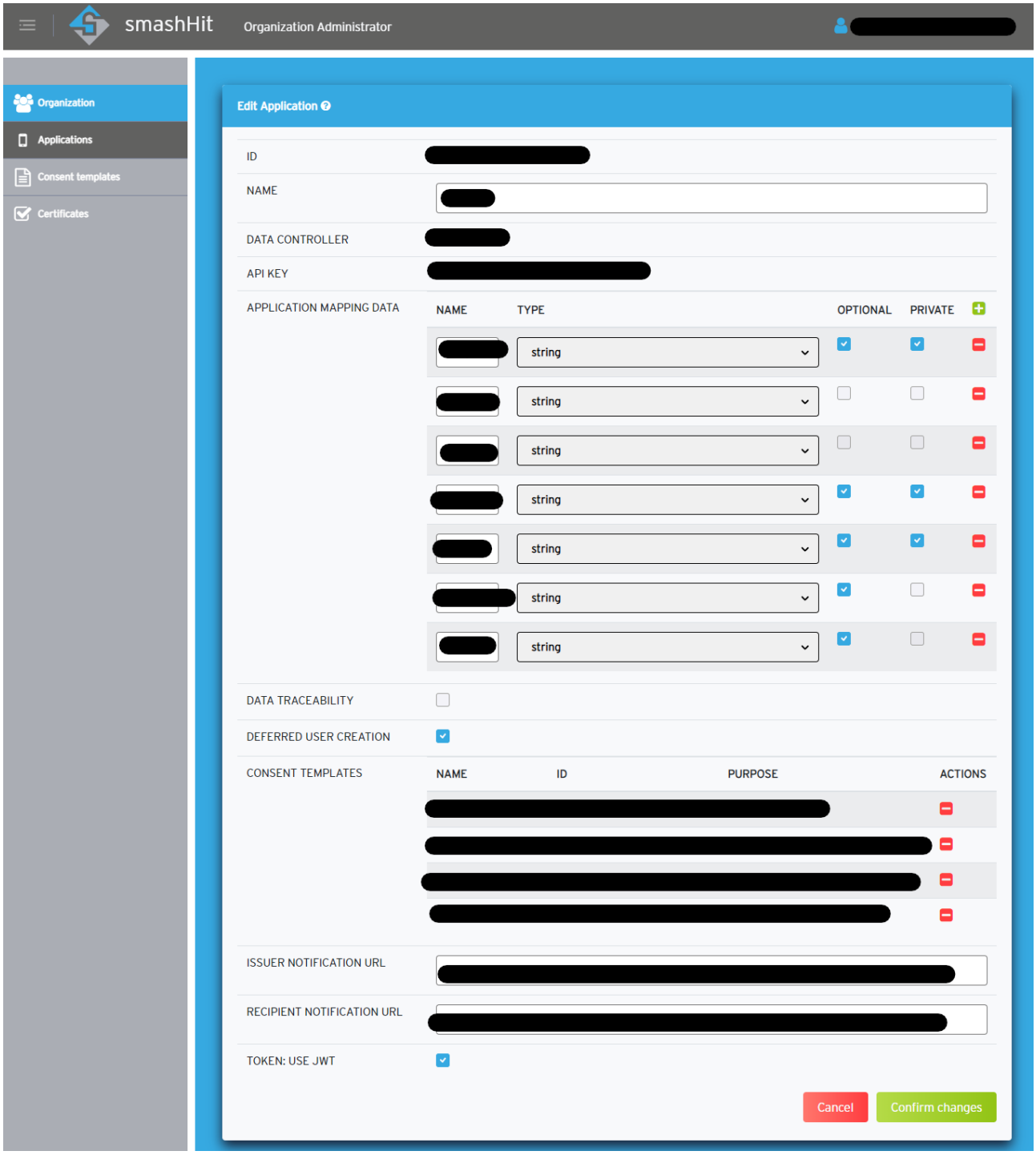


Figure 7: Application registration

As it can be seen from the figure above, in order to model an application, an organization must provide:

Field name	Description
Name	A common name to identify the application.
Application mapping data	Set of custom fields that the application developer can declare for the application.

	If defined, the application can send an “applicationMappings” object in the certificate request setting these values, so that smashHit will store and distribute these fields in future notifications about the consent certificate. The purpose of setting these fields is to facilitate the integration and mapping smashHit objects (consent details, application or data subject ids…) with internal ids used by the organization, for example, a vehicle identification number, or product ID. Fields flagged as “private” will only be forwarded to the certificate issuer application but hidden from other recipients of the consent.
Data traceability	Whether to enable or disable the traceability capabilities for the application[3].
Deferred user creation	If enabled, the application can set a personalData.email object in the consent “/certificate” request so that smashHit will create a new account for the dataSubject using the email provided (if it doesn’t exist already). If disabled, the application must manually register the data owner through a POST request to the /dataOwner endpoint available in the smashHit API[4].
Consent templates	All the consents to be requested by an organization’s application must follow one of the consent templates specified here. See the next section for details on the consent templates.
Issuer/recipient notification url	The url where smashHit will send a notification about new consents or changes to existing consent in which this application is involved. See the notification section for more details.
Token: use JWT	Whether the notification request from smashHit should include a bearer or JWT token[5].

Once the application is registered, please note that smashHit will assign it an “api key” that can be consulted from the application detail view. Please take note of this value as it will be used later for certificate generation.

Consent template management

Consents in smashHit must always follow a custom, pre-defined template. Similar to applications, consent templates can be listed (Figure 8), managed (Figure 9) and defined (Figure 10, Figure 11) from the “Consent templates” tab.

[3] Depending on the use case, the organization should define a specific consent template (and ask the user for consent) regarding the processing of data for traceability purposes.
[4] https://smashhit.ari-mobility.eu/swagger/
[5] Please contact smashHit staff to obtain the JWT secret for token validation

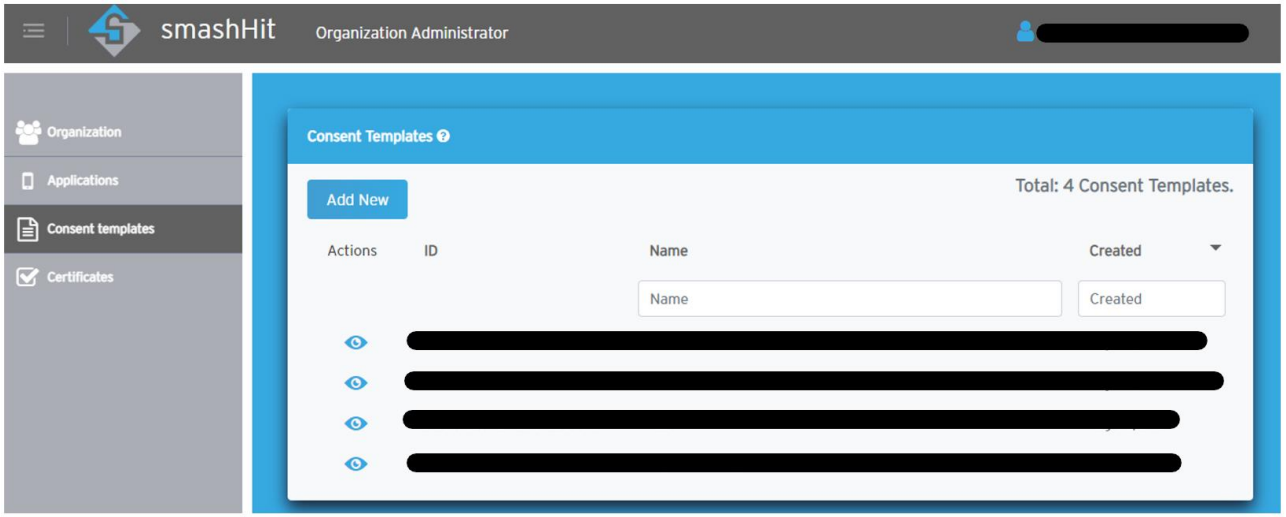


Figure 8: Consent template list

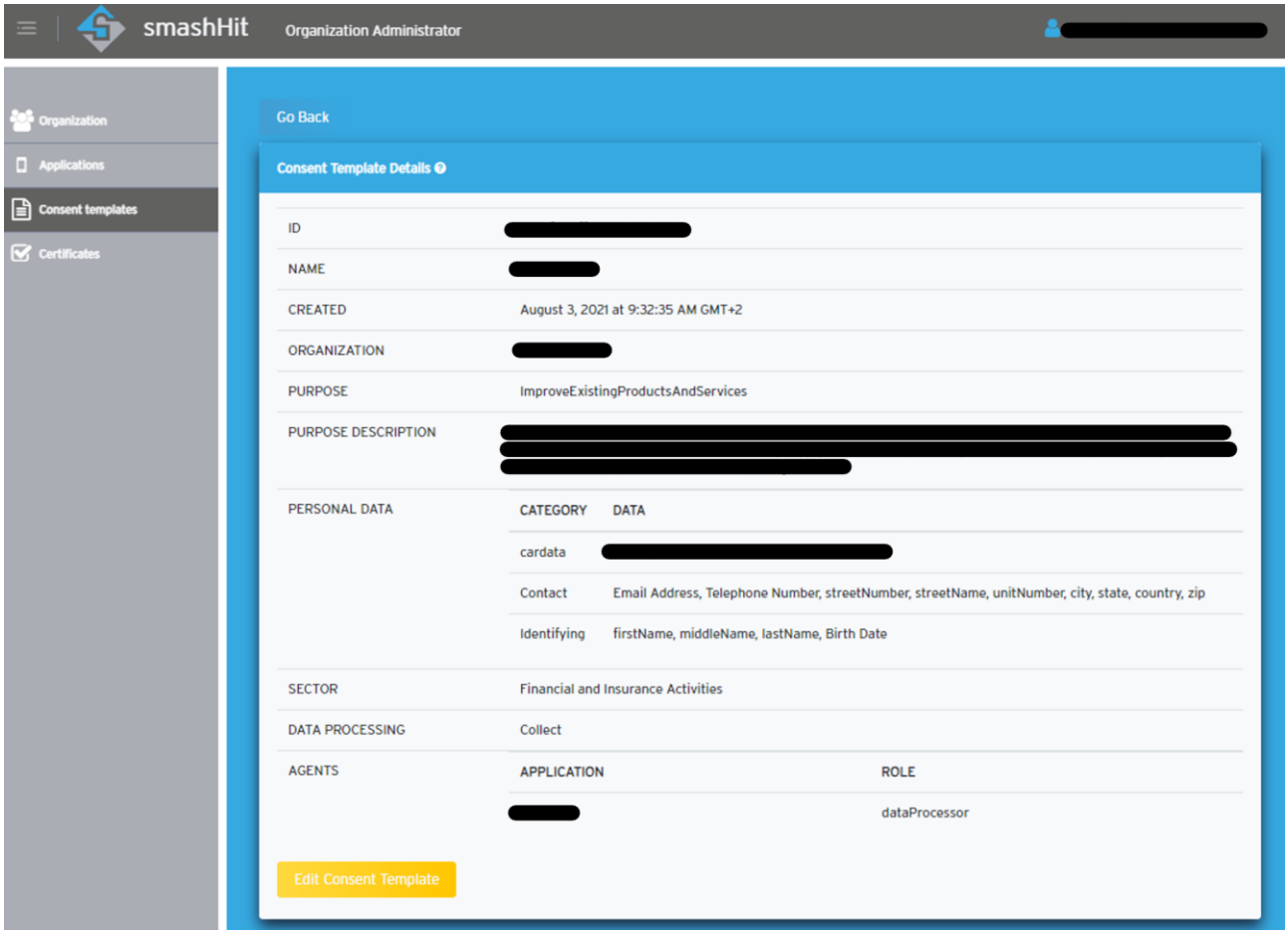


Figure 9: Consent template management

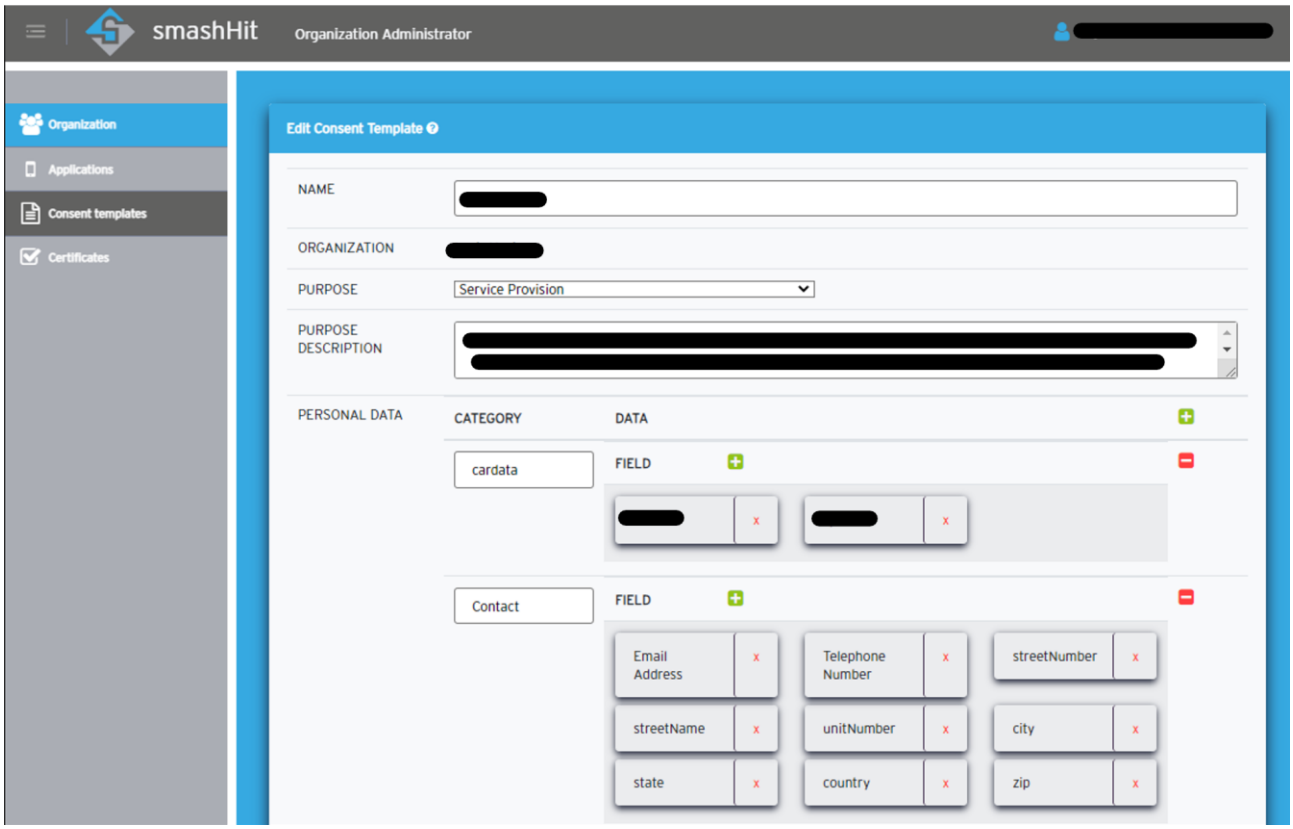


Figure 10: Consent template definition (I)

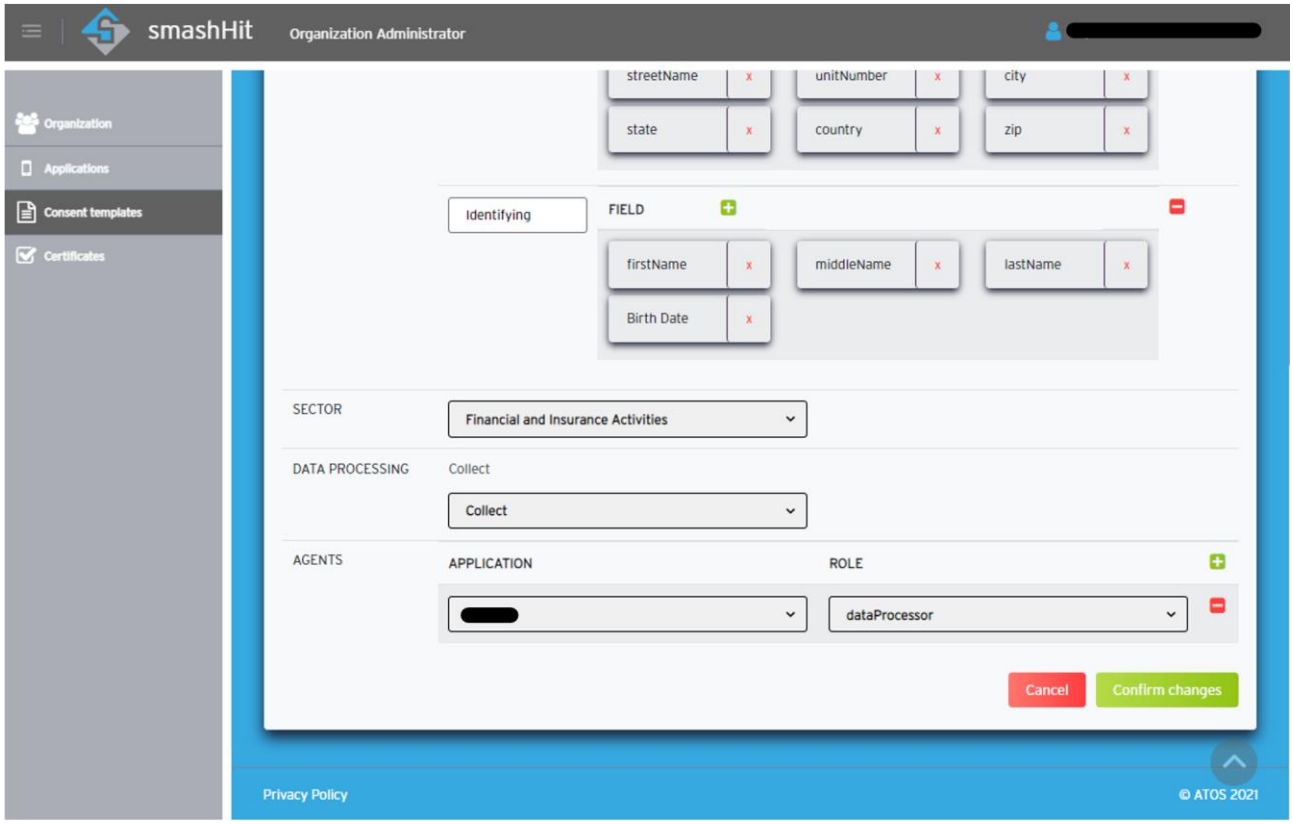


Figure 11: Consent template definition (II)

As it can be seen from the figures above, the following fields must be defined for each consent template. The formulary is guided/restricted by the smashHitCore Ontology[6] (for more details see the smashHit semantic model technical essay on the smashHit website at https://smashhit.eu/publications), please refer to the appropriate field in the ontology for a detailed definition and list of possible values.

Field name	Description
Name	A common name to identify the consent template.
Purpose	The purpose of the processing of personal data.
Purpose description	Optional, detailed description of the purpose.
Personal data	List of types of personal data involved in the consent.
Sector	Business sector applicable to the processing.
Data processing	List of processing operations to perform with the personal data.
Agents	List referencing each of the organizations involved in the consent as well as their role in the processing of personal data. The organization is referenced through its relevant application, which will be notified of the creation/modification of consents following this template.

Similar to applications, consent templates are assigned an ID by smashHit as they are created. Please take note of this value as it will be used later for certificate creation.

Consent management

After this initial setup of the organization, application, and consent templates, organizations can start generating consent “certificates” through the smashHit.

In the context of smashHit, a consent “certificate” is a JSON object which aggregates a group of consents, requested by the same organization (through one of its applications) to a given data subject. After asking the data subject for a set of consents through its application, following the list of previously defined consent templates, the organization sends smashHit a request to “certify” the consents.

In short, this “certification” step consists of smashHit internally validating, signing, storing, and distributing the consents to Agents involved.

While the previous steps can be achieved through calls to the smashHit API directly, it is recommended to use the GUI as described. The next steps however do require the integration of some calls to the smashHit APIs.

Following industry standards, the smashHit APIs require an oauth2 token for authorization (to be included in each request as an “x-auth-token” header). In the case of smashHit, this is managed through Keyrock IdM[7], so an access token can be generated through a standard oauth2 procedure[8] using your organization user credentials and the client_id and client_secret provided by the smashHit team, as the example seen in Figure 12.

[6] https://smashhiteu.github.io/smashHitCore/
[7] Deployed at: https://idm-smashhit.ari-mobility.eu
[8] See: https://fiware-idm.readthedocs.io/en/7.4.0/api/

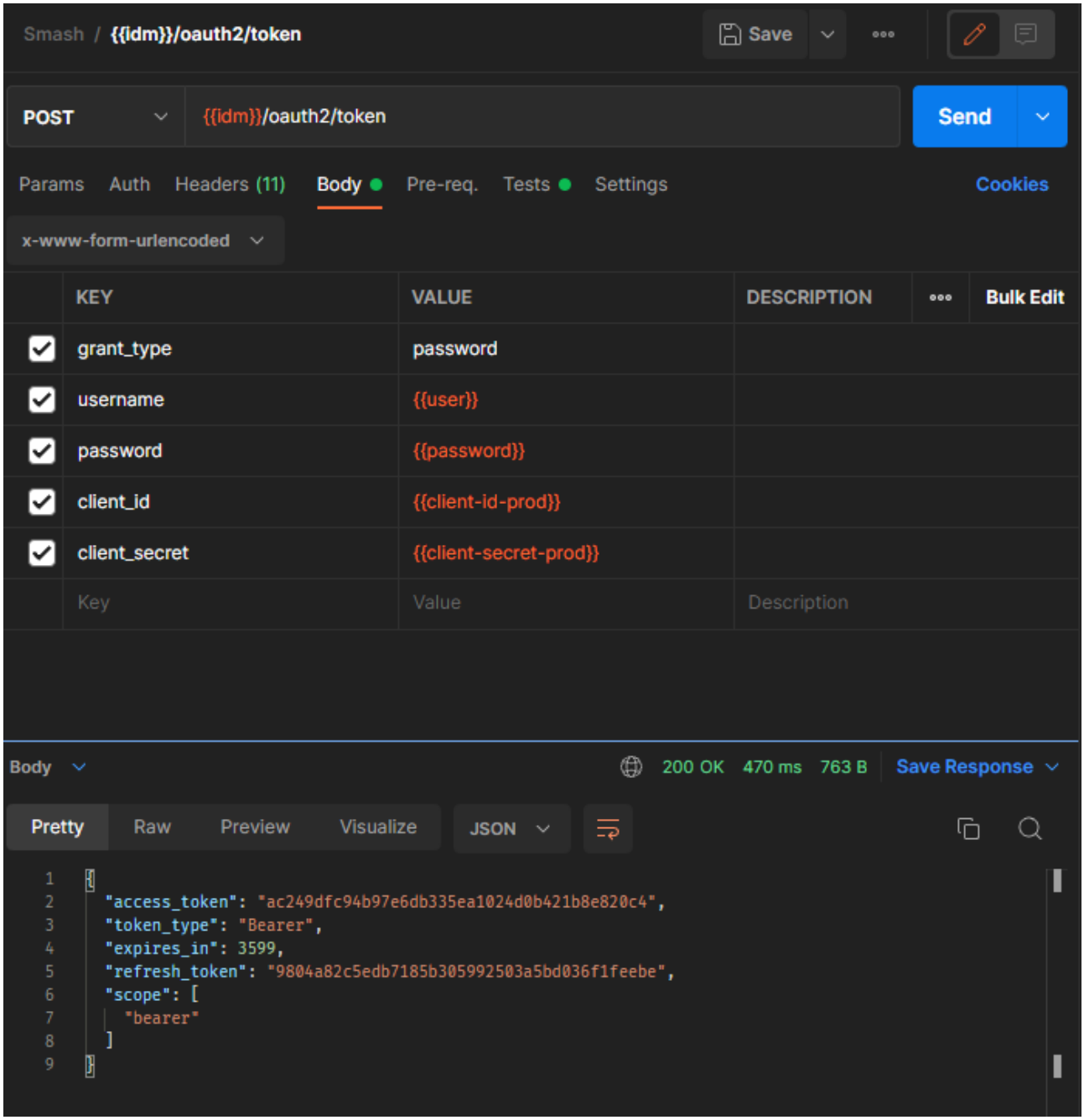


Figure 12: Access token generation

Once the organization has configured its application and its consent templates, the application can start registering new data subjects (users with data owner role) and consent “certificates” in smashHit. As mentioned earlier, applications can either:

- Manage data owner registration manually from the smashHit API (/dataOwner endpoints), in which case they must set the “dataSubject” field in the certificate request as the dataOwner’s idmId field
- Enable the “deferredUserCreation” option for implicit registration in the certificate request itself, in which case the certificate request can include a “personalData” object with an “email” attribute, which the platform will use for registration.

So, with either a registered dataSubject id or a new email, the application's api key (to be set in the "x-api-key" header), and the consent template ids, the organization can already build certificate requests following the POST /Certificate endpoint definition, as seen in Figure 13.

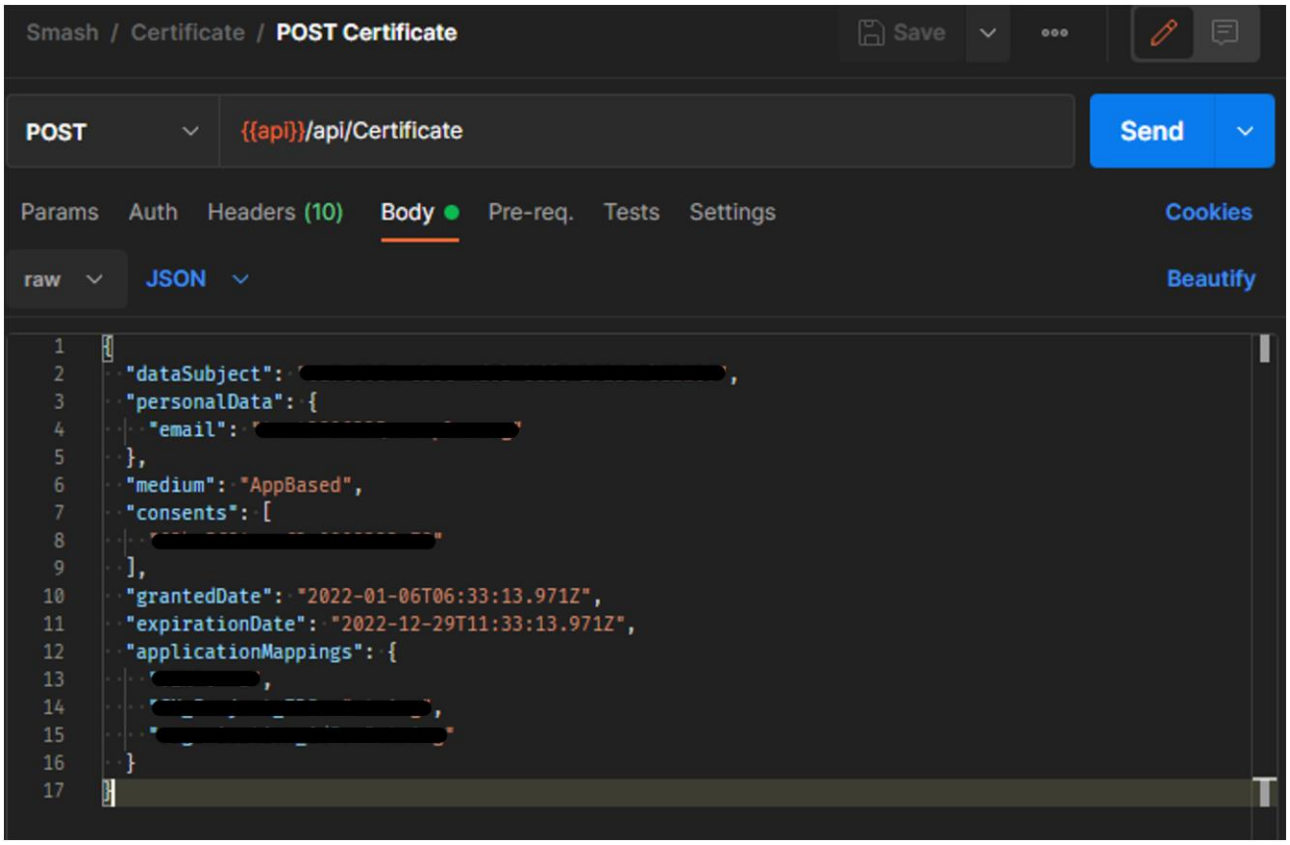


Figure 13: Certificate request - Postman example

As it can be seen, the request must include:

- Either the dataSubject id or an email if it requires registration
- The medium, grantedDate & expirationDate of the consent
- The applicationMappings
- The consents field itself. This field must be an array containing only the ids of those consentTemplates agreed by the user.

smashHit will process this certificate request, and transparently handle the necessary grant/revoke requests with the ACT in the background, and respond with the final consent certificate, indicating the current status of the consents involved in this certificate. In the example request above, if everything is correct, the organization can expect back a certificate with only the consent modelled by the template specified in the "consents" field granted. The other consent templates tied to his application will be considered as not accepted by the user, and thus will be either in a "NOT_GRANTED" or "REVOKED" (if they had been granted in the past) status. For more details, please check the full definition in the swagger specification.

Once the consent certificate has been registered in smashHit, data subjects can use their credentials to access the GUI, check the consents, and most importantly, modify them, in which case smashHit would issue a notification to the appropriate applications so the new consent status can be enforced.

In case the data subject asks the organization directly (outside smashHit) to change its consent preferences, resulting in the revoke or grant of any consent, the organization's application must send

again a new certificate request with the updated user selection to smashHit. smashHit will automatically handle any revocation/grant request to reflect the changes in the ACT and respond to the request with a new certificate reflecting the new changes.

Any processing errors will be returned in a “consentErrors” field. In case any error makes it impossible for smashHit to apply the new changes in the ACT’s knowledge graph, the consent will remain unchanged, while the error will be reported back to the application.

While certificates can also be queried using smashHit API (check the specification to find different methods of querying/filtering by dataOwner, application, etc.), the smashHit GUI does provide an organization interface to check the active certificate for each user, in the “Certificates” tab (Figure 14).

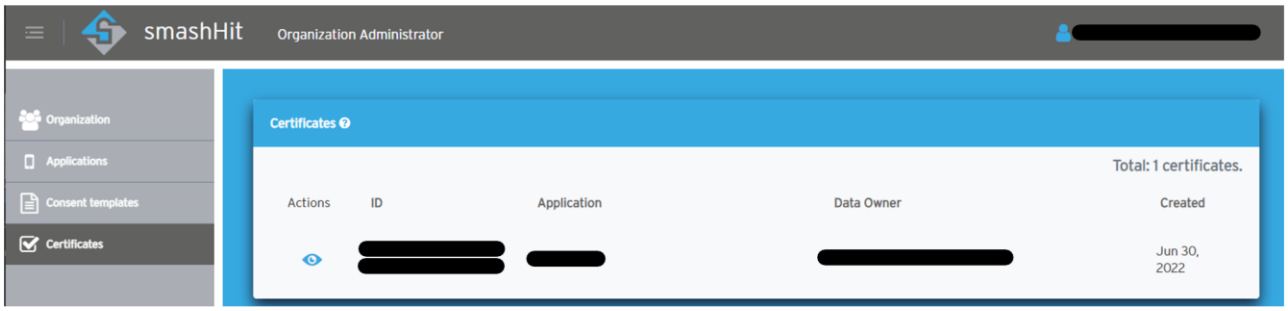


Figure 14: Certificates list

Organization users can also click one of the certificates to check the full details, including the status of each of the consents in the certificate (Figure 15), as well as the traceability records for a consent, assuming it is enabled for the application (Figure 16).

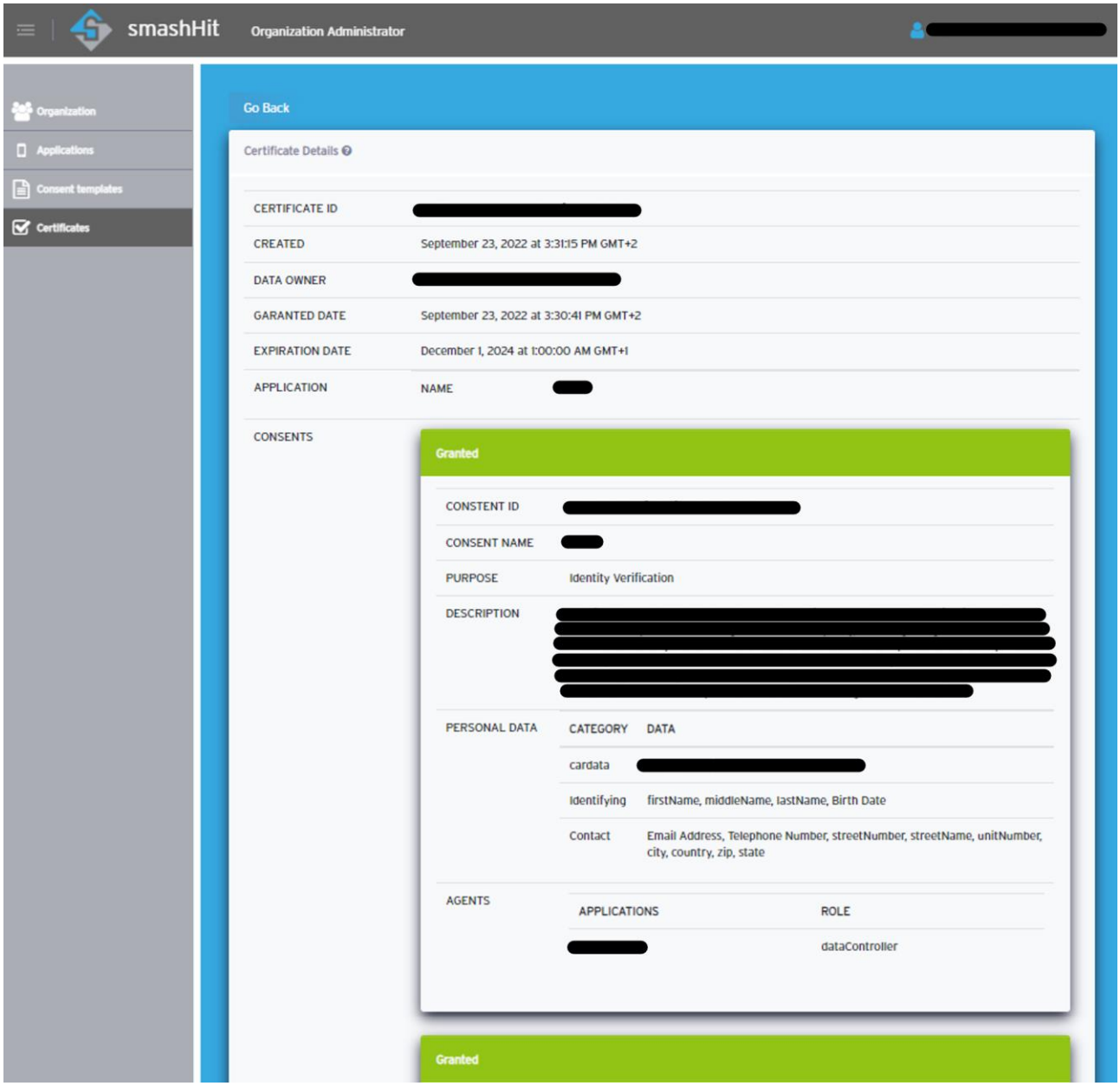


Figure 15: Certificate details

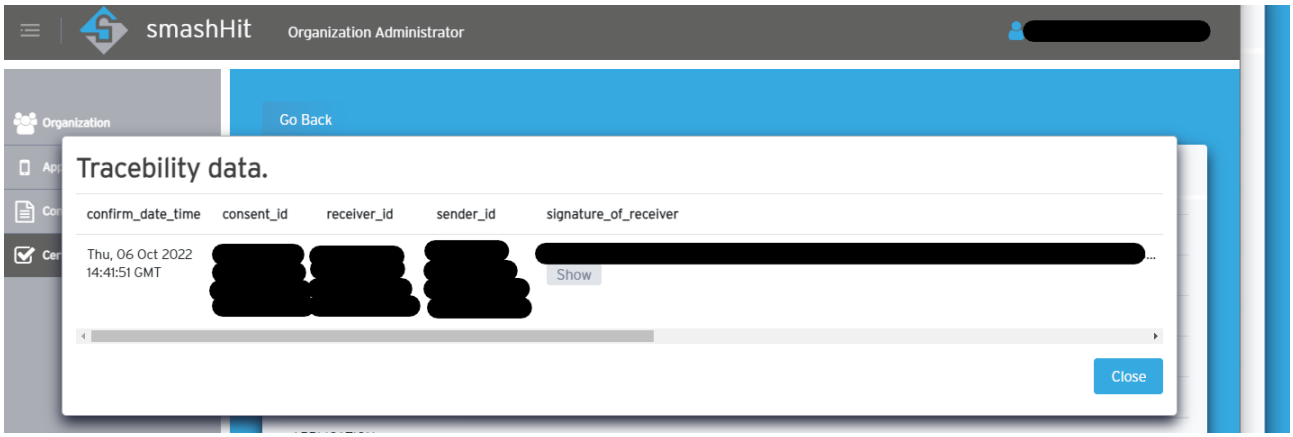


Figure 16: Consent traceability records

Notification endpoint

The previous section covers certificate management from point of view of the issuer application, i.e., the application which originally collects the consent from the user and handles the certificate.

However, as it has been mentioned earlier, the rest of the agents of a given consent must be notified of events affecting new and existing consents in smashHit as long as they are involved in the processing.

Even for issuer applications, it is necessary to implement some kind of notification channel as well, so that when a data subject uses the smashHit GUI to modify their consents, their application can be informed and enforce the new preferences.

As described in the Application management section earlier, organizations can configure a notification URL for their applications, so that smashHit can automatically inform them of any changes to an existing consent (or the creation of a new one, in the case of third parties).

This notification URL must follow the swagger specification defined here:

https://app.swaggerhub.com/apis/icarrilloari1/smashNotification/0.0.4

As it can be seen from the specification, the process is simple: smashHit will send a POST request to the URL configured with the new certificate as the body (including a reference to the previous certificate, as well as the latest applicationMappings). As the notification is received by an application, the new consent selection must be enforced.

Data use traceability

This section is to help organization developers to use the Data Use traceability (DUT) component. The DUT component has two independent parts which are: Data Traceability and Re-identification. Concerning Data Traceability, the actions that could be done are: register data, initiate a data transfer, confirm an occurred transfer, get the data trace using consent or contract identifier. Concerning the re-identification part, the implemented actions aim to verify the authenticity of a leaked data (verify if a given data was modified or not) in two scenarios: First, the data subject or provider does not modify the original data, and that data is leaked and modified by an attacker. Second, the data subject modifies the original data by inserting watermark and by creating a watermarked data; this watermarked data is leaked and modified by an attacker. The actions are: re-identify fingerprint, watermark trip and find watermarked correlation.

Before going into further details, we are first going to explain how the DUT component is set up and made ready to use. As prerequisites, Python 3 has to be installed. There are two possibilities to set up DUT component which are traceability package and docker.

- Traceability package: download the code here[9], extract the folder, install the required dependencies (present in requirements.txt) in your working environment, then follow the step 4 mentioned in this file[10].

[9] https://github.com/uttam1216/DataUseTraceabilty_LUH_UBO
[10] https://github.com/uttam1216/DataUseTraceabilty_LUH_UBO/blob/main/README.md

- Docker: install docker[11] (if not yet present), download the code here[12], extract the folder and follow step 5 mentioned in this file[13].

After these previous setting steps, concluded by the onboard of your company (running the "main.py" file), you exist as an entity inside the DUT component. By opening the link http://localhost:5001/swagger-ui/ on the server where DUT is being installed, you have access to all the functionalities offered by the Data Use Traceability Component. For each of those functionalities, here is a short description.

1. **Hashing**: get the hash of any data. It changes with a small change in the data. The input parameter is
    a. **Trip_data***: a JSON file with trajectory coordinates of the trip.

    The output will contain:

    b. **Hash**: hash of the data to be used for register data and other APIs.

2. **Fingerprinting**: fingerprints the initial raw data. The fingerprint of the data does not change with a small change in the data. The input parameter is:
    a. **Trip_data***: a JSON file with trajectory coordinates of the trip.

    The output will contain:

    b. **fingerprint**: fingerprint of the data to be used for register data and other APIs.

3. **Register Data**: register data (information about data that needs to be sent to another partner later) into the centralised manager to have a trace. During registration, following inputs are used:
    a. **access_token***: security token allowing the company to use the service.
    b. **consent_id***: identifier of the consent given by the data subject, to save and exchange the data.
    c. **contract_id**: identifier of the contract against the consent that is given by the data subject, to save and exchange the data.
    d. **creation_time**: time at which the company receiving the information from the subject, collected that data.
    e. **expiration_time**: time up to which all the companies in possession of that data are allowed to use it, as defined by contract with the data subject.
    f. **hash***: hash of the pure information without any metadata.
    g. **fingerprint**: fingerprint of the pure information without any metadata.
    h. **watermark**: fingerprint of the pure information without any metadata.

[11] https://docs.docker.com/get-docker/
[12] https://github.com/uttam1216/DataUseTraceabilty_LUH_UBO/tree/main/company_module_docker
[13] https://github.com/uttam1216/DataUseTraceabilty_LUH_UBO/blob/main/company_module_docker/README.md

i. **origin**: id of the data subject.

The output of this API will contain:

j. **uri**: internal smashHit identifier of the registered data that will be used as input of other APIs like Notify Data Transfer and Confirm Data Transfer.
k. **error:** In case of any error, the message appears as a response.

4. **Notify Data Transfer**: notify the manager about an upcoming data transfer from one company to another. To trigger the function, the sending company needs:
   a. **access_token***: security token allowing the company to use the service.
   b. **uri***: uniform resource identifier of the data, created by the manager during the registration process.
   c. **receiver_name***: smashHit registered name of the company or organisation which will receive the data

   The output of this API will contain:

   d. **sender_id**: smashHit registered id of the sender to be given as input of Confirm Data Transfer API
   e. **signature_of_sender:** unique signature of the sender to be given as input of Confirm Data Transfer API
   f. **message:** success or failure message

   Note: To receive the parameters and the actual data, the receiver can create a web service (collection of APIs) where endpoints for getting parameters and data could be place. Then he might send the URL to the sender. In this case the sender will have to call those 2 endpoints just after “Data transfer”.

5. **Confirm Data Transfe**r: acknowledge receiving of data. The parameters of this function are:
   a. **access_token***: security token allowing the company to use the service.
   b. **hash***: hash of the data or metadata, created upon the data registration phase.
   c. **sender_id***: public identifier of the company sending the data.
   d. **signature_of_sender***: signature of hash made by the sender module.
   e. **uri***: uniform resource identifier of the data created by the manager during the registration phase.

   The output of this API will contain:

   f. **message:** success or failure acknowledgement message

6. **Consent Trace**: obtain log of data transfers for a given **consent_id**.

   The output of this API will contain:

   a. Data Trace w.r.t. consent in JSON format **OR** a message if no trace exists.

7. **Contract Trace**: obtain log of data transfers for a given **contract_id**.

The output of this API will contain:

a. Data Trace w.r.t. contract in JSON format **OR** a message if no trace exists.

8. **Re-Identify Fingerprint:** verify authenticity of data for the scenario in which the data owner or provider does not insert a watermark beforehand. The input is
   a. A JSON formatted body with key names “own_data” and “trajectory”, where trajectory is the possibly leaked and modified trajectory and own_data is the data set with which the data owner wants to compare given trajectory.

   Internally the Model creates a “fingerprint” or, more specifically, a vector representation of the trajectories given. This fingerprint is modification invariant, thus small modifications of the trajectory will not change the fingerprint. Further the model compares the fingerprints of the trajectory and own data set and finally predicts whether the trajectory is a modified version of a trajectory in own data.

   Output:

   b. **prediction*:** True if the trajectory is a modified Version of a trajectory in own_data, false otherwise

9. **Watermark Trip:** insert watermark to create a watermarked Trip. The input to watermarking should be a trip data (JSON as in the sample file provided).

   Output & Intermediate Files:

   a. Output is a watermarked trip in json format in the same as the input format. The difference between original trip and watermarked trip is that the original latitude-longitude pairs have been replaced by watermarked latitude-longitude pairs.

   Input files, intermediate files and output files are saved with the corresponding names as a backup inside the company module (wtrace_data folder).

10. **Watermarked Correlation:** compute correlation to verify authenticity of data for the scenario in which the data owner or provider modifies the original data. The input to this API should be a
    a. **watermarked_trip*:** a JSON file with trajectory coordinates of the trip for which we want to assess the authenticity.
    b. **trip_id*:** id of the original trip data that was leaked.

    The outputs are:

    c. **correlation**: a number between 0 and 1. It represents the correlation between the original watermark and the watermark that is extracted from the watermarked data (with/without intruder’s attack).
    d. **message**: the categorical description what the correlation value means

(*): Mandatory input parameters

In addition to what has been mentioned above in this section, there are some standalone modules developed, using data generation mechanisms, geolocation embeddings, concept learning approaches and semantic enrichment of spatiotemporal data, to connect data traceability and re-identification, and perform post processing analysis. Some of those components are already available in the GitHub repository[14] and others are in the initial stage of development. For available components, all details about installation and setup are contained inside the README.md file of the component.

Automatic contracting

This developer guide helps the organization developers to make use of the contract module. The ACT contracts module supports the management of contracts and compliance verification checks on contracts via REST APIs and a contract UI. For better API documentation, we use the Swagger[15]. We can access the contract APIs endpoints from here:

https://actool.contract.sti2.at/swagger-ui.

The contract API endpoints divide into the following eight parts:

(i) Contracts.
(ii) Company.
(iii) Contractors.
(iv) Users.
(v) Term Types.
(vi) Contract Terms.
(vii) Contractor Signatures.
(viii) Contract Obligations (clauses).

Each part has CRUD functionality for the management of contracts. As an example, we can create a term type via the following endpoint:

https://actool.contract.sti2.at/contract/term/type/create/

To update a term type, the endpoint below can be used:

https://actool.contract.sti2.at/contract/term/type/update/

And to access all the term types we can use the endpoint:

https://actool.contract.sti2.at/contract/term/types/

All the endpoints have similar functionality. For a better understanding of the use of contract APIs, we provide the general steps for creating a contract:

1. Create all the term types, which require for contracts.
2. Add contractual parties.
3. Create a contract B2B/B2C with basic information (e.g., type of contract, start and ending date, etc.).

[14] https://github.com/D-Stiv/smashHitUBO.git
[15] https://swagger.io/solutions/api-documentation/

4. Add signatures to the contractor signatures with contract id, which is created in step 3.
5. Create contract terms with contract id, which is created in step 3.
6. Create contract obligations with (contract id, contractor id, term id), which are created in step 3, 2, 5.
7. Update the contract with contractors, contract terms, contract obligations, contract signatures.

The contract UI is another way to interact with ACT contract module. The UI can be accessed from here: https://stiinnsbruck.github.io/smashHit-UI/build/web/index.html#/contract/.

Guidelines for testing via UI:

1. Login form requires a contractor id, which can be generated during the registration process and can be viewed from the contract rest endpoint (https://actool.contract.sti2.at/swagger-ui).

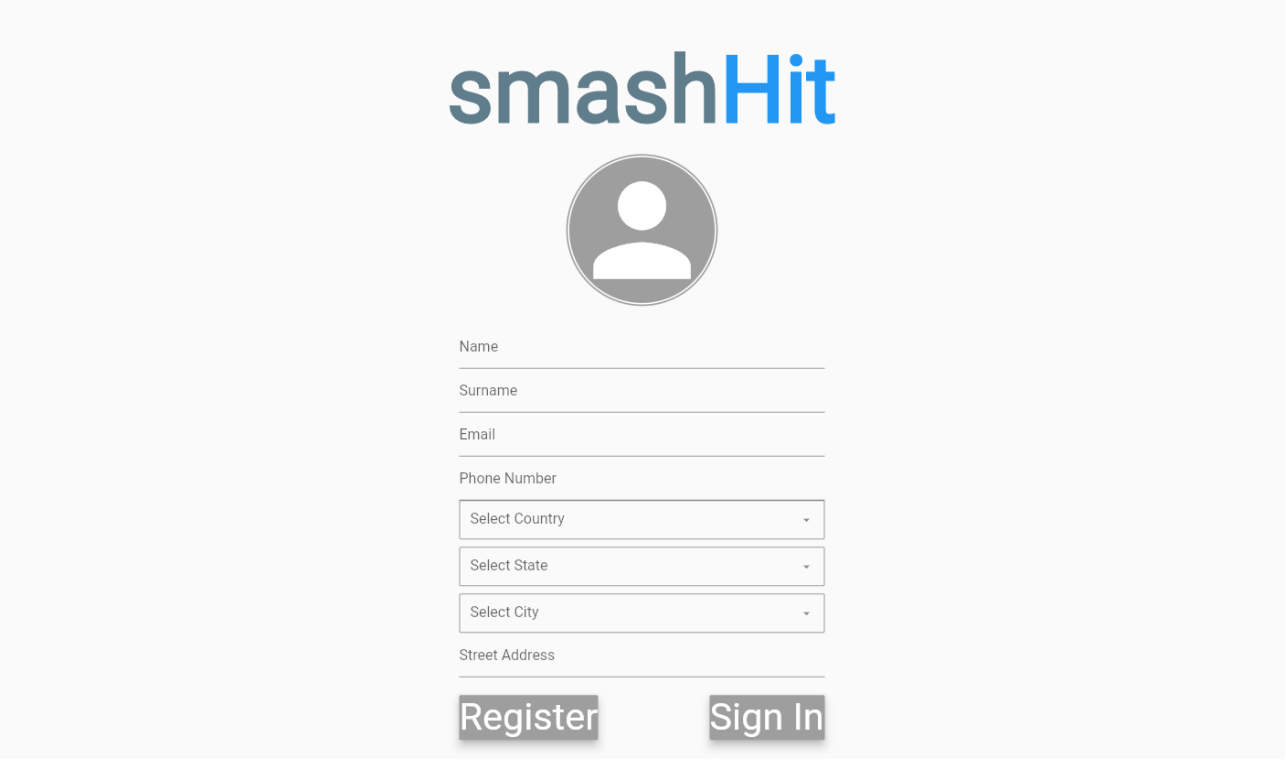


Figure 17: Contract UI landing page

2. You can use the menu icon to perform different functionalities, such as creating new contract term types, creating a new contract, viewing the contractor profile, dashboard, etc.

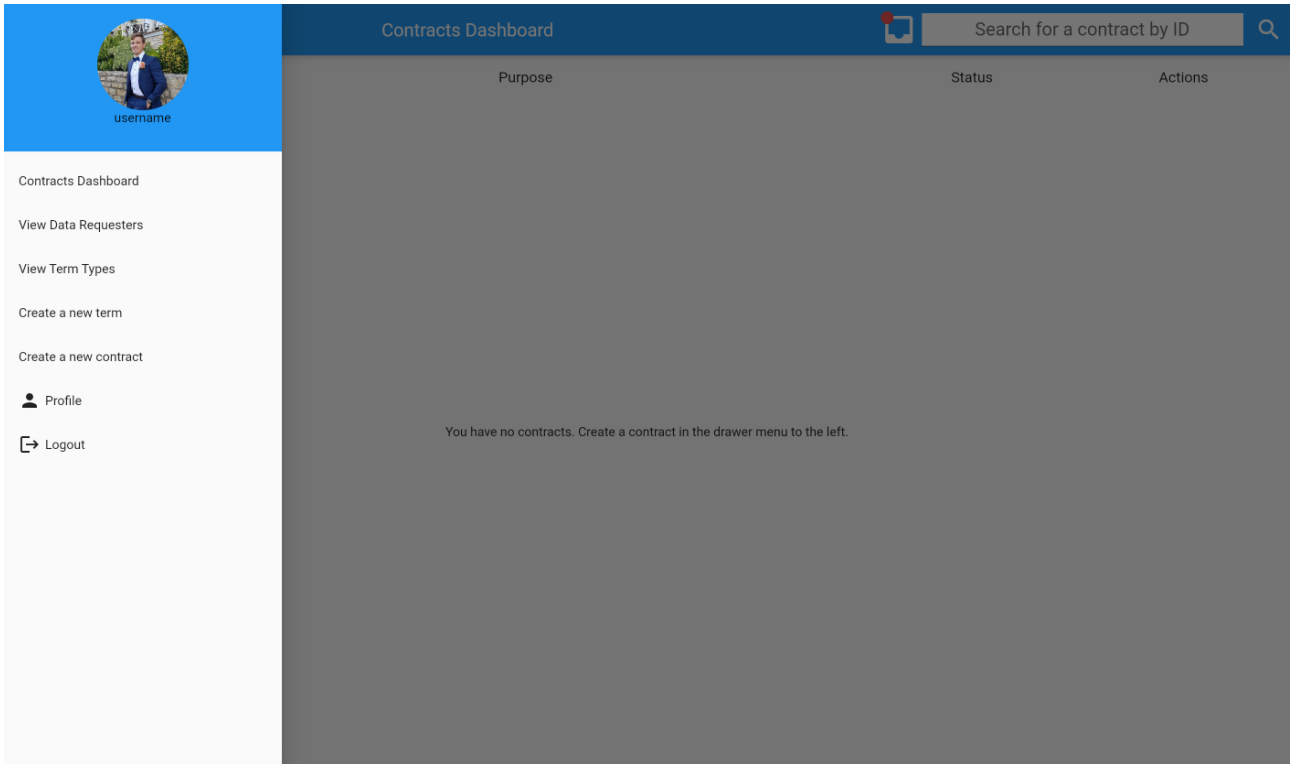


Figure 18: Contract UI menu

3. With this GUI you can create different categories of contracts, such as a business-to-consumer contract, a business-to-business contract

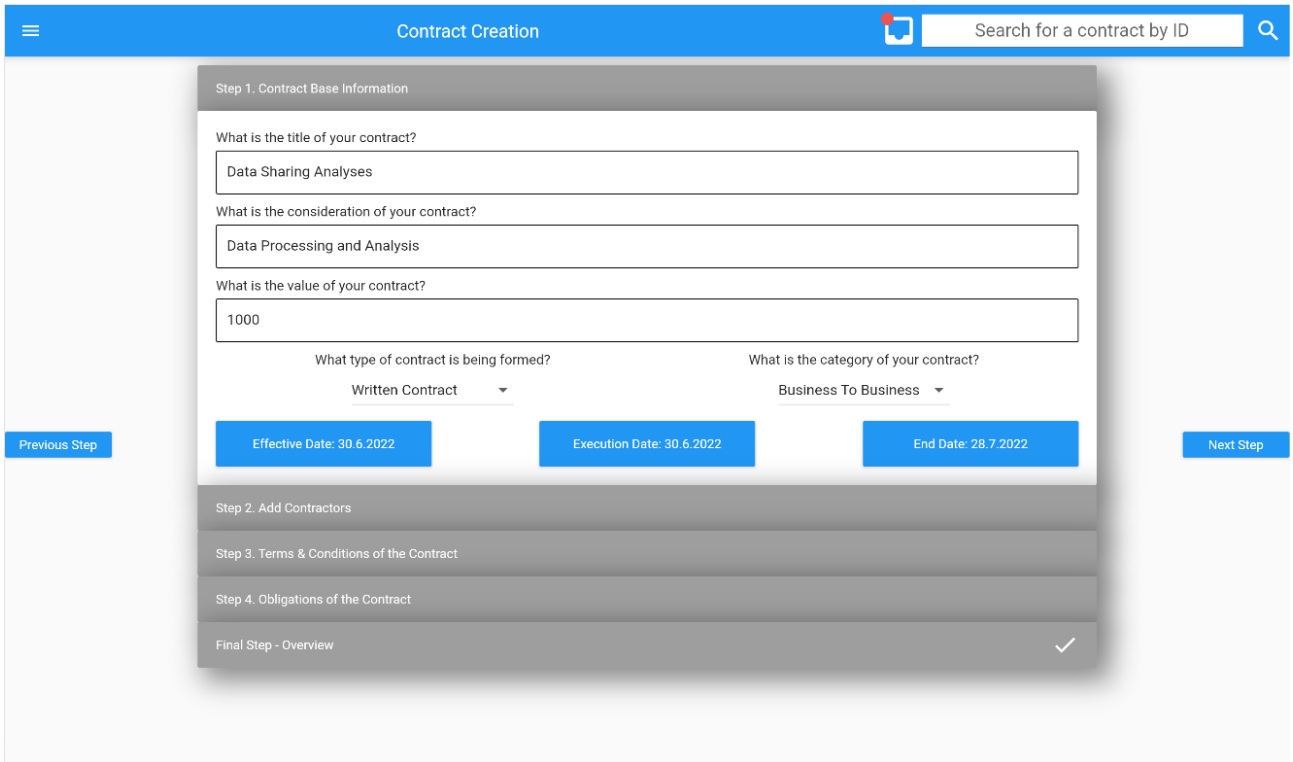


Figure 19: Contract creation

The BC developer can also get the source code from the link below and run locally:

https://github.com/AmarTauqeer/Contract

The steps need to be performed:

1. git clone https://github.com/AmarTauqeer/Contract.git
2. go to the backend folder
3. update the .env file
4. sudo docker-compose -f docker-compose.yml up

Finally, the following list comprises the tools and technologies required for the contracting module to run locally:

1. Flask
2. Flask-RESTful
3. uwsgi
4. flask_apispec
5. SPARQLWrapper
6. unittest
7. Docker

Context-Sensitive Security and Privacy

This guide to the context-sensitive security & privacy policies and mechanisms is for developers of other smashHit framework components as well for data provider/processor organization user developers. The S&P mechanisms consists of three main components: Policy Tool, Policy Server and Event Processing Point. These components provide support for the creation, testing, and deployment of S&P policies in the smashHit platform via REST APIs and an interactive command-driven interface. The APIs are documented using Swagger specifications which are published at:

https://app.swaggerhub.com/apis/tog-rtd/ngac-apis/1.1.1.

The defined endpoints comprise the following categories[16]:

(i) Policy administration – allows the building and manipulation of policies stored in the Policy Server.

(ii) Policy query – allows a query to be posed to the policy decision procedures relative to a policy currently stored in the Policy Server, typically resulting in a “grant” or “deny” response.

(iii) Event-response policy – used to report events and to configure, using the Event-Response Language (ERL), responses consisting of a sequence of commands to be performed when an associated event pattern is recognized.

(iv) Security auditing – used to configure the audit system, including the selection of a set of audit events to be recorded in the audit log if and when they occur, and management of the audit log files.

[16] There are several additional categories of API endpoints defined in the referenced Swagger specification that are TOG-RTD APIs and not used in smashHit Automatic Contracting tool.

(v) DPLP meta-element administration – permit meta-elements defined in the Declarative Policy Language (DPL) extended for privacy (DPLP) to be added to or deleted from a policy stored in the Policy Server. These meta-elements create a GDPR-compliant abstraction layer on top of the primitive DPL policy elements.

Categories (i) policy administration, (ii) policy query and (v) DPLP meta-element administration are used by the smashHit automatic contracting tool, and may be used by other smashHit components or other BC applications in future developments. Category (iv) security auditing, is used by the Policy Server internally, and the external API is provided for use by smashHit data use traceability. The value of (iv) to smashHit generally can be enhanced as other components define security-relevant events that they can recognize and subsequently participate in the reporting of occurrences of these events through the security auditing API.

The Policy Tool provides an interactive environment in which to develop and test DPL, DPLP and ERL policies. The Tool does not provide policy file editing capabilities although it does provide the capability of incrementally building a policy by adding policy elements. The typical workflow for the Policy Tool is in conjunction with a text editor in another window. When the policy is changed in the editor and saved[17], the policy file can be reloaded into the Tool for continued testing. Many of the operations available through the Server's APIs are also available as commands in the Tool. The commands and APIs are elaborated in the smashHit deliverables: D4.2 Final Specification of S&P Policies and Mechanisms, and Privacy Metrics, Section 5; and, D4.4 Full Prototype of S&P Mechanisms and Privacy Metrics, Section 5.

To test policies or applications that use NGAC through the REST APIs it is easy to build and run a local instance of the Policy Server and Event Processing Point (EPP).[18] In this way testing can be done without interference with the production instance of the server. smashHit and organization developers can obtain the source for the S&P mechanisms at:

https://github.com/tog-rtd/SmashHit.git.

The source repository consists of a directory tree including source files and example files. Building and running the S&P mechanisms requires an installation of SWI Prolog (preferably a current or recent version) which is available for several operating environments, including Mac, Windows and Linux from:

https://www.swi-prolog.org.

The executables for the Policy Tool and the Policy Server are built by executing the simple script 'mkngac' in the BUILD directory (e.g. "BUILD/mkngac"), which will create the 'ngac' and 'ngac-server' executables in the BUILD directory. These are built to include the Prolog runtime environment and can be executed like ordinary executables.

After running the mkngac script the Policy Tool can be started at the OS command level by executing "BUILD/ngac". The Policy Tool has some built-in self-tests. These can be run to ensure that the Policy Tool is working correctly. The commands of the Policy Tool are in the syntax of a function call, with the command name, an optional list of parameters enclosed in parentheses, and a terminating period. The command "self_test." will run the built-in self-tests. "help." will list the available commands. "help(command)." will give more detailed help on the named command. Command

[17] Some text editors, e.g. Visual Studio Code, can continuously auto save.
[18] By default, the EPP is started and runs in the same process as the Policy Server. Command line options for the ngac-server are listed in D4.2, Section 5.5.1.

sequences may be packaged as predefined procedures (“procs”) by editing the procs.pl file and executed with the “proc” command, but they can more conveniently be created as command scripts in separate files and executed with the “script” command.

The Policy Server can be started at the OS command level by executing “BUILD/ngac-server”.[19] The Policy Server has some command line options that are useful when starting it in a production environment. For example, the setting of the admin token for this run of the Server. The admin token is needed for all calls to the policy administration APIs. The smashHit startup procedure should distribute to components that use the APIs the chosen value for the token given to the current instance of the Server. For convenience the default token value “admin_token” can be used on local instances of the Server that are used for testing API-based interactions.

Tests that demonstrate the use of the Server’s APIs are contained in the source repository under the TEST/nn-tests directory, where may be found a number of shell command scripts that use the curl utility to invoke the APIs. These scripts send a sequence of requests to the server to test for known correct answers. These scripts run illustrative tests, they do not run exhaustive tests. A top-level script, run-nn-tests.sh, runs all of the numbered test scripts. Expected results are contained in corresponding numbered files. The script takes a single optional argument “-json” which causes the results of the run to be compared against the numbered files with “-json” suffix. The ngac-server must be running in the appropriate mode (--jsonresp the default) to get JSON responses. (See server command line options.)

An alternative method of gaining familiarity with the APIs is by using SwaggerHub to invoke the APIs[20] which are defined in a Swagger specification at the link given above.

Global parameters are defined in the param.pl file. Settable parameters (those that can be changed from the Policy Tool command line with the set command) are itemized in the list settable_params. For example, the parameter audit_logging is a settable parameter.

In production use smashHit components, such as the ACT, that use the Policy Server will need to build the policies in the Server using the Server’s APIs. It is the responsibility of the using application to assure that the version of a policy in the Server is current. In the event that the Server is shut down the using application will have to reestablish the policy in the appropriate state. The Server offers an API (“paapi/readpol”) to read out the current state of a policy and an API (“paapi/reset”) to reset a policy to its state when it was initially created. These APIs may be useful in managing the current state of policies in the Server. The developer of the using application(s) will need to work out a strategy for policy administration and coordinate with the platform administrator to have appropriate procedures for startup and shutdown of smashHit to implement this strategy. (See also: Platform Administrator User Guide.)

[19] It is possible to start the Server from within the Policy Tool for testing but the command line options are not available when starting it this way.

[20] The APIs can be invoked on a locally running instance of the Server or on an instance running at ATB by editing the “host” lines near the beginning of the Swagger specification.

F.A.Q.

smashHit platform

Q: What is the smashHit platform?

A: smashHit is a platform that provides a generic, transparent way to view and manage consent certificates supported by disrupting technologies such as traceability of use of data, data fingerprinting and automatic contracting among the data owners, data provider, and service providers.

Q: How can my organization keep track of new consents or modifications to existing ones?

A: Regardless of if you are the issuer of the consent or just an agent involved in the processing, whenever a consent status is registered in smashHit, or if the data subject changes their decision (grant/revoke), smashHit will automatically send a notification to your application via the URL provided in the configuration. Please check the “Notification endpoint” section for more details on the format.

Q: Do consent notifications requests from smashHit include any security measures?

A: Outbound notification requests from smashHit will include an ouath2 token issued by the smashHit IdM. In the same way that smashHit requires inbound requests to provide a valid authorization token, your application should expect and validate this token in the x-auth-token of any incoming request.

Q: How should I notify smashHit in case the data subject asks to revoke a given consent?

A: Any modifications to an existing consent can be processed simply by sending a new consent certificate request with the current status as selected by the data subject. The new consent certificate will replace the previous one.

Q: In the consent certificate request, how can I specify whether a consent is accepted or not by the data subject?

A: Only the id for currently accepted consent templates must be included in the certificate request (in the “consents” array), regardless of if these have changed from previous requests.

Since smashHit is already aware of which consent templates are linked to a given application, there is no need to specify consents not accepted by the data subject.

Also, since smashHit is already aware of the previous certificate, there is no need to specify changes in the request. smashHit will automatically compare the current data subject selection with the previous certificate’s and handle any necessary consent grant or revocation.

In case a specific consent has not changed its status from a request to the next one, its id and decision token will maintain their values.

Q: How can I register my users with smashHit?

A: Applications can either:

- Manage data owner registration manually from the smashHit API (/dataOwner endpoints), in which case they must set the “dataSubject” field in the certificate request as the dataOwner’s idmId field
- Enable the “deferredUserCreation” option for implicit registration in the certificate request itself, in which case the certificate request can include a “personalData” object with an “email” attribute, which the platform will use for registration.

Glossary

ACT: Automatic Contracting Tool

Agent: Entity that bears some form of responsibility in the context of a consent

Administrator: smashHit platform system administrator

Consent: As per Article 4(11) of the GDPR, 'consent' of the data subject means any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her

Consent template: A template defining the details of a data processing by an Agent (smashHit organization user), such as the purpose of the processing or the personal data to be processed

CSS: Context sensitivity solution

Data owner: smashHit user role which acts as the data subject according to GDPR

DPL: Declarative Policy Language

DUT: Data Use Traceability

ERL: Event response language

GDPR: Abbreviation for 'General Data Protection Regulation', a legal norm on EU level adopted in 2016, which is directly applicable within its scope and lays down rules for the processing of personal data so as to protect natural persons' fundamental rights and freedoms, in particular their right to the protection of personal data

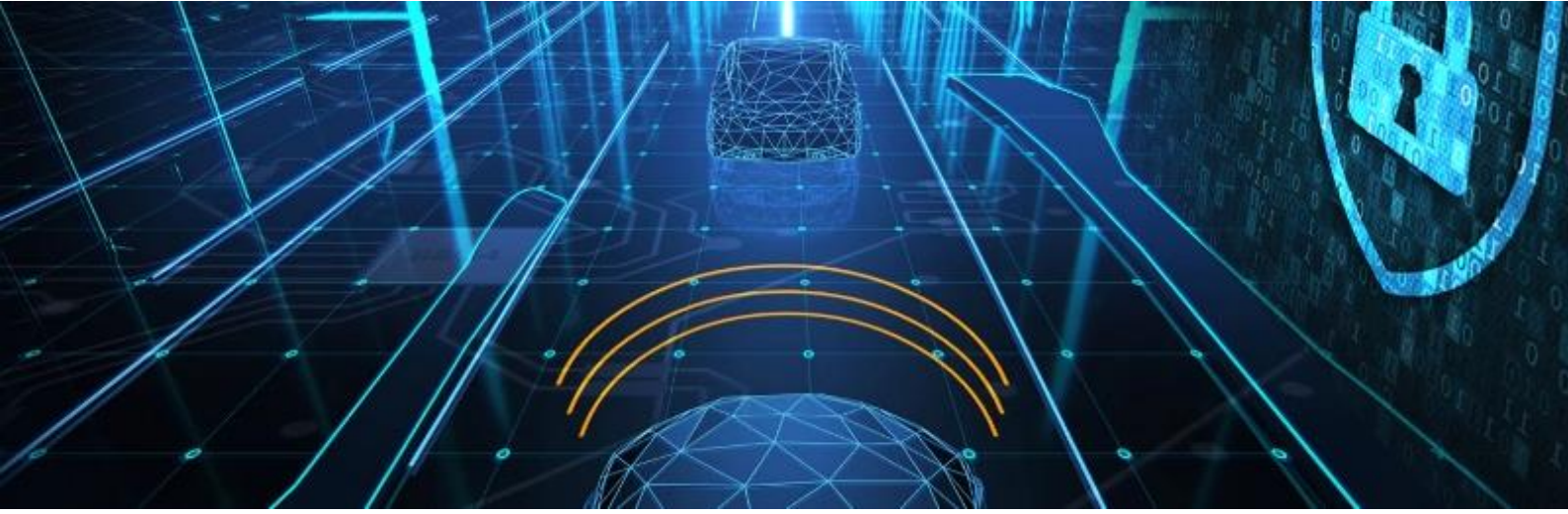
Personal data: Any information which are related to an identified or identifiable natural person (Art. 4 (1) GDPR), e.g. a name or a home address

Purpose: The purpose of Data Handling/Processing

S&P: Security and privacy module

Figures

Figure 1: Organization registration 4
Figure 2: Sign in 5
Figure 3: Organization landing page 5
Figure 4: Organization management 6
Figure 5: Applications list 7
Figure 6: Application management 7
Figure 7: Application registration 8
Figure 8: Consent template list 10
Figure 9: Consent template management 10
Figure 10: Consent template definition (I) 11
Figure 11: Consent template definition (II) 11
Figure 12: Access token generation 13
Figure 13: Certificate request - Postman example 14
Figure 14: Certificates list 15
Figure 15: Certificate details 16
Figure 16: Consent traceability records 16
Figure 17: Contract UI landing page 22
Figure 18: Contract UI menu 23
Figure 19: Contract creation 23



smashHit

Our vision - Solving Consumer Consent & Data Security for Connected Car and Smart City

Further information

This document is part of the smashHit Methodology. The complete set of documents, including other user/developer guides as well as white papers created within this scope can be found on our website:

https://smashhit.eu/publications

Our consortium

ATB

universität innsbruck
Department of Computer Science

Leibniz Universität Hannover

LexisNexis RISK SOLUTIONS

THE Open GROUP

infoTripla

UNIVERSITÄT BONN

Atos

FORUM VIRIUM HELSINKI

Funded by the Horizon 2020 Framework Programme of the European Union