assistants
Testing per smart home sicure, private e sostenibili

D2.3

Data Privacy Dashboard

WP2 – Smart Home Dashboard

ASSISTANTS

Assessing Software Security, prIvacy, and SusTAiNability through Testing techniqueS

Due date of deliverable: 31/03/2025 (M16)
Actual submission date: 31/03/2025

Responsible partner: Politecnico di Torino (POLITO)
Editor: Luca Ardito
E-mail address: luca.ardito@polito.it

28/02/2026
Version 1.1

Project funded by the European Union – NextGenerationEU within the PRIN 2022 PNRR programme		
Dissemination Level		
PU	Public	X
CO	Confidential, only for members of the consortium	



The ASSISTANTS project (P2022WEAH7, CUP E53D23016400001) is funded by the European Union – NextGenerationEU under the PRIN 2022 PNRR programme of the Italian Ministry of University and Research (D.D. 1409 of 14/09/2022), Mission 4, Component 2, Investment 1.1.

Authors: Luca Ardito (POLITO)

Approved by: Luca Ardito (POLITO), Principal Investigator

Revision History

Version	Date	Name	Partner	Sections Affected / Comments
1.0	31/03/2025	Luca Ardito	POLITO	All. Final version
1.1	28/02/2026	Luca Ardito	POLITO	All. Revision for publication on the project website

Disclaimer

Funded by the European Union – NextGenerationEU. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

Executive Summary

This deliverable accompanies the release of the **Data Privacy Dashboard**, the software result of Task T2.3 of the ASSISTANTS project. The dashboard is a GDPR-oriented web platform through which the people living in a smart home, the parties responsible for the applications that process their data, and the Data Protection Officers manage consent, privacy documentation and privacy rules. The rules are propagated to the devices of the home.

The software is released as open source under the MIT licence in the repository `Assistants-Project/Privacy-Dashboard`. Its main capabilities are:

- one interface for the three GDPR roles, with role-specific views: Data Subject, Data Controller and Data Protection Officer (DPO). Controllers and DPOs are bound to the applications for which they are responsible;
- giving and withdrawing consent per application and per home, and exercising the GDPR rights of access, portability, erasure, restriction, complaint, removal of data and consent withdrawal;
- privacy notices, GDPR compliance questionnaires and privacy impact assessments, all editable and exportable as PDF;
- privacy rules written in a high-level language, translated into XACML by the Policy Translation Point and verified for conflicts with Semantic Coloured Petri Nets;
- automatic enforcement: consent decisions and user rules become `privacy_rule` topics in the Distributed Hash Table (DHT) of the home-automation platform, where they are applied to the devices;
- synchronisation of homes, rooms, devices, applications and rules with the home-automation platform, and registration of applications by a marketplace.

The system is a React/TypeScript single-page application on top of a Flask REST back end with PostgreSQL. It exposes 74 REST endpoints in 14 blueprints with OpenAPI documentation and is verified by an automated API test suite of 404 requests and 1,264 assertions (Postman/Newman); a reduced collection of 161 requests and 317 assertions runs in the continuous integration pipeline. It is deployed with Docker.

Table of contents

Executive Summary 3
1 Introduction 6
1.1 Purpose and scope 6
1.2 Relation with other deliverables 6
1.3 Background 6
2 Software Release 7
2.1 Repository and licence 7
2.2 Repository structure 7
2.3 Technologies 7
3 Architecture 8
3.1 Overview 8
3.2 Back end 8
3.3 Front end 9
3.4 Authentication and authorisation 9
3.5 Data model 9
4 Features 10
4.1 Data Subject 10
4.2 Data Controller 10
4.3 Data Protection Officer 10
4.4 Marketplace and home-automation platform 11
5 Policy Translation Point and Conflict Detection 12
5.1 Rule language 12
5.2 XACML translation 12
5.3 Conflict detection with Semantic Coloured Petri Nets 12
6 Enforcement Pipeline and Integration with the Home-Automation Platform 14
6.1 Synchronisation 14
6.2 Propagation flow 14
6.3 Device-level rules 14
6.4 Distributed state 15
7 Testing, Continuous Integration and Deployment 16
7.1 Testing 16
7.2 Continuous integration and delivery 16
7.3 Deployment 16
8 Operational Metrics 17

9 Installation and Use 19
9.1 Prerequisites 19
9.2 Configuration 19
9.3 Demonstration mode 19
9.4 Development mode 19
9.5 Testing mode and regression suite 19
10 Conclusions 20
References 21
Glossary 22

1 Introduction

1.1 Purpose and scope

The third objective of ASSISTANTS is to let users manage their data and their GDPR rights from a single point, with their preferences propagated automatically to the smart-home devices. D2.1 defined the requirements and design of the dashboard and D2.2 the enforcement tools. T2.3 integrates these lines of work into a working platform.

This deliverable is of type **Software**. The software itself is the repository described in Section 2. This document describes:

- where the software is and how it is licensed;
- the architecture and the technologies used;
- the features offered to each role;
- the Policy Translation Point and the conflict detection;
- the enforcement pipeline and the integration with the home-automation platform;
- testing, continuous integration and deployment;
- the operational metrics measured on the system;
- installation and use.

1.2 Relation with other deliverables

- **D2.1** provided requirements, use cases and design.
- **D2.2** provided the Usage Control tools and the device-level enforcement chain that the dashboard feeds.
- **D3.2** uses the dashboard as the policy front end of the pilot home.
- **D3.3** validates the dashboard within the pilot.

1.3 Background

The dashboard builds on the results of the SIFIS-Home project, where the core of a privacy dashboard and the Policy Translation Point were first developed [1]. Within ASSISTANTS the system was consolidated into a single platform and connected to a real home-automation system. As a result, the policies declared by users are actually enforced on devices.

2 Software Release

2.1 Repository and licence

Item	Value
Repository	https://github.com/Assistants-Project/Privacy-Dashboard
Licence	MIT
Integrated release	branch `frontend` (back end, front end, Docker deployment, documentation, benchmarks)
Documentation	`README.md`, `docs/` (requirements document, Smartotum integration guide, architecture, data-flow and entity–relationship diagrams), Swagger/OpenAPI at run time

The repository also keeps the development branches of the back end and of the integration with the home-automation platform, so that the history of the components is preserved.

2.2 Repository structure

Path	Content
`backend/app/controllers`	REST controllers (14 blueprints)
`backend/app/services`	Business logic: consents, rules, XACML translation, Petri-net analysis, DHT, notifications
`backend/app/repositories`, `models`, `schemas`	Data access, ORM entities, request/response schemas
`backend/app/integrations/smartotum`	Client of the home-automation platform (authentication, homes, DHT topics)
`backend/app/ontologies`	OWL ontologies used by the Policy Translation Point
`backend/testing`	Mock data of the home-automation platform and Postman API test collections
`frontend/src`	React/TypeScript application: pages, components, contexts, API clients
`docker`	Dockerfile, demo compose file and CI scripts
`scripts/benchmarks`	Reproducible performance benchmarks
`docs`	Requirements, integration guide and diagrams

2.3 Technologies

Layer	Technologies
Front end	React 18, TypeScript, Vite, React Router, React-Bootstrap, AG Grid, rich-text editor for notices, PDF rendering for exports
Back end	Python 3.11, Flask 3, flask-smorest (OpenAPI), marshmallow, SQLAlchemy 2 with Alembic migrations, Flask-JWT-Extended, PyJWT (JWKS verification)
Policies	XACML 3.0 generation, OWL ontologies (Owlready2), Semantic Coloured Petri Nets
Data	PostgreSQL
Deployment	Docker, Docker Compose, GitLab CI

3 Architecture

3.1 Overview

The dashboard follows a three-tier architecture (Figure 1). Only the back end interacts with the DHT of the home-automation platform, to read the topology of the home and to apply rule changes; the front end reaches it exclusively through the REST API. The Policy Translation Point (PTP) is a separate module, invoked by the back end when privacy rules are committed. It translates them into XACML and verifies them.

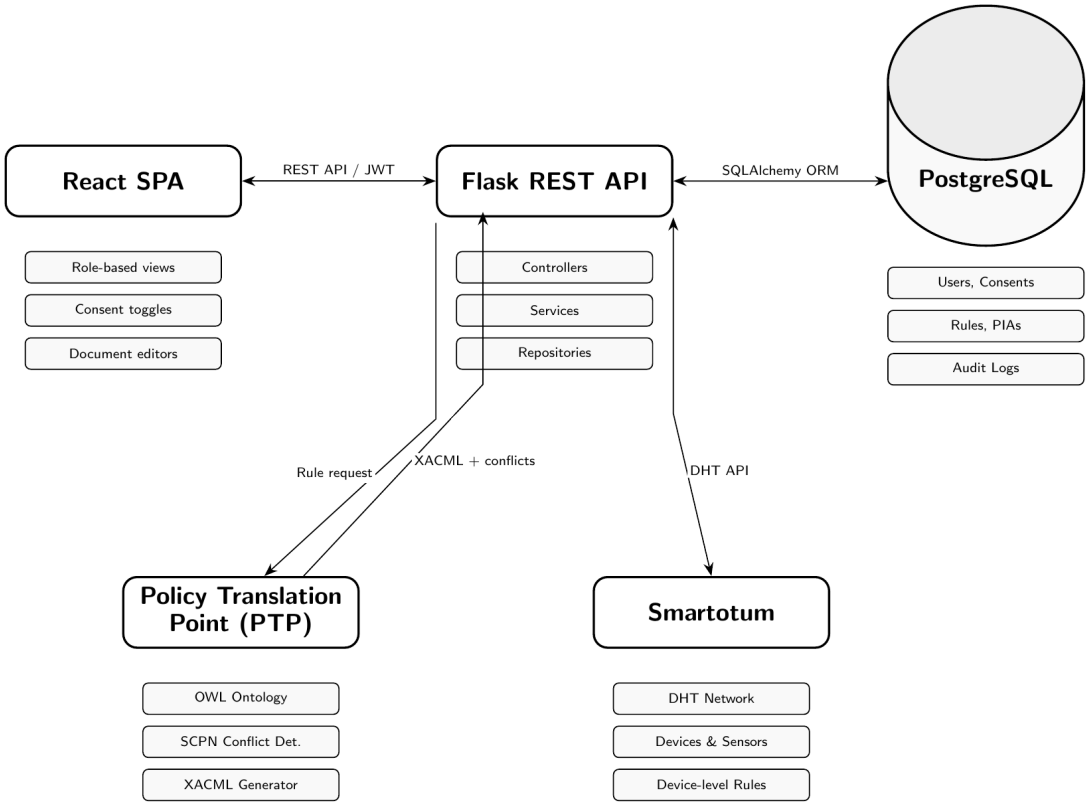


Figure 1: High-level architecture of the Data Privacy Dashboard

3.2 Back end

The back end follows the Controller–Service–Repository pattern:

- **Repositories** wrap all database access for each entity and contain no business rule.
- **Services** contain the business logic and coordinate repositories and external clients. For example, a consent change locates the affected rules and propagates them to the home.
- **Controllers** expose the REST endpoints through blueprints that generate OpenAPI documentation automatically.

Dependencies flow strictly downward, and services emit domain events that are captured by the audit trail.

The REST API is organised in 14 blueprints: applications, authentication, consents, DHT, homes, integrations (marketplace), messages, notifications, privacy impact assessments, privacy notices, questionnaires, rights requests, policy translation and users. Together they expose 74 endpoints.

3.3 Front end

The front end is a single-page application with role-based navigation. Three shared contexts hold the authentication state, the selected home and the installed applications. The interface applies progressive disclosure: a Data Subject sees installed applications, consent toggles and the buttons to exercise rights; Data Controllers and DPOs additionally see editors, request trackers and reporting pages. The main pages are:

- applications and consents;
- rights requests;
- messages and notifications;
- privacy notices, questionnaires and privacy impact assessments;
- the policy translation editor;
- the privacy overview of the home.

3.4 Authentication and authorisation

- **Data Controllers and DPOs** log in with credentials managed by the dashboard; passwords are stored hashed.
- **Data Subjects** log in with their account on the home-automation platform. The back end forwards the credentials to the platform and verifies the returned token against the platform's public key set (JWKS). It then checks that the token subject matches a registered user.
- **Token handling.** Tokens are kept in HTTP-only cookies. The credentials of the home-automation platform are never stored by the dashboard.
- **Role checks.** Role-based access control is enforced on each endpoint through decorators that inspect the token.

3.5 Data model

The data model covers the following entities:

- users and their roles;
- smart homes, rooms and devices synchronised from the home-automation platform;
- applications with their consent catalogue, owner Data Controller, manager Data Controllers and DPO;
- consents per Data Subject, application and home;
- privacy notices, questionnaires and privacy impact assessments based on the template of the French data protection authority (CNIL);
- privacy rules with their XACML translation and deployment status;
- rights requests, messages and notifications.

Changes to consents, rules and notices are recorded in audit tables, and notifications keep a snapshot of the information needed for accountability.

4 Features

4.1 Data Subject

- Select one of the homes the user has access to, and see the applications installed there that process personal data, with description, Data Controllers, DPO and privacy notice.
- Give consents from the catalogue of each application. Withdraw some or all consents through a dedicated request (Figure 2). The system automatically creates the corresponding deny rules and enforces them on the devices.
- Exercise GDPR rights through requests: access, additional information, portability, erasure, restriction of processing, complaint, removal of all personal data, consent withdrawal. Follow their status.
- Define high-level privacy rules for a device, a room or the whole home.
- Read and download the privacy notices of the applications.
- Exchange messages with the Data Controllers and DPOs of the applications, and receive notifications.

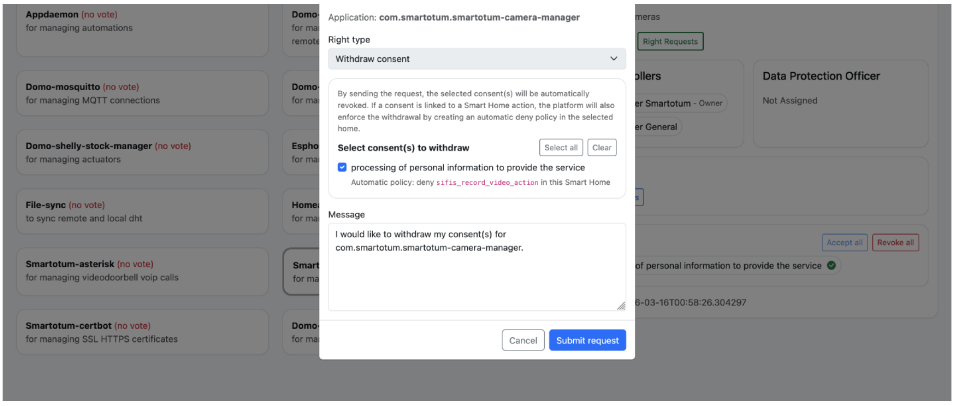


Figure 2: Consent withdrawal: the Data Subject selects the consents to withdraw and the system creates and enforces the corresponding deny rules

4.2 Data Controller

- See and manage the applications for which they are owner or manager, including the applications registered by the marketplace.
- Write privacy notices from scratch or from a template in a rich-text editor, associate them with applications and export them as PDF.
- Fill in GDPR compliance questionnaires. Applications are colour-coded by result, and a summary highlights the critical aspects.
- Carry out privacy impact assessments with a guided form and export them as PDF.
- Receive, answer and close the rights requests of Data Subjects, without seeing the concrete configuration of their homes.
- Add manager Data Controllers to an application and assign its DPO.

4.3 Data Protection Officer

A DPO has privileges similar to those of a Data Controller on the applications assigned to them; each application has at most one DPO. The DPO follows requests, documentation and consents of these applications, and so can monitor their compliance.

4.4 Marketplace and home-automation platform

- A marketplace registers an application through a dedicated endpoint protected by an integration token. The request names the owner Data Controller, optional delegates and the catalogue of available consents. A dedicated update operation reconciles metadata and consent catalogue.
- The applications installed in the homes are imported from the home-automation platform, together with their consent catalogue. A consent can be mapped to a device action, which is denied automatically while the consent is not given.

5 Policy Translation Point and Conflict Detection

5.1 Rule language

Rules are written in a structured, almost natural form:

- **subject:** a person, a role or everyone;
- **action:** for example recording video or detecting motion;
- **resource:** a device or capability;
- **effect:** allow or deny;
- **pre-conditions:** temporal or contextual;
- **post-conditions:** logging or notification.

The editor builds the rule from the triggers and actions available in the home. These are derived from the devices synchronised from the home-automation platform and described by an OWL ontology that the dashboard regenerates for each home [2].

5.2 XACML translation

The PTP translates each rule into an XACML 3.0 policy [3]. The policy target matches subject and action, conditions become XACML conditions on environment attributes such as the current time, and rules are combined with *deny-overrides*. The generated policy can be inspected and downloaded from the interface.

5.3 Conflict detection with Semantic Coloured Petri Nets

Before deployment, the rules of a home are verified with Semantic Coloured Petri Nets [4]. Device states are modelled as places and rules as transitions; tokens carry subject, action, resource and time constraints (Figure 3). Simulating the net reveals two kinds of conflict:

- **inconsistencies:** opposite effects on the same action and resource;
- **redundancies:** identical effects that can be merged.

The verification report lists the conflicting pairs, and the responsible user resolves them before the rules reach the devices (Figure 4).

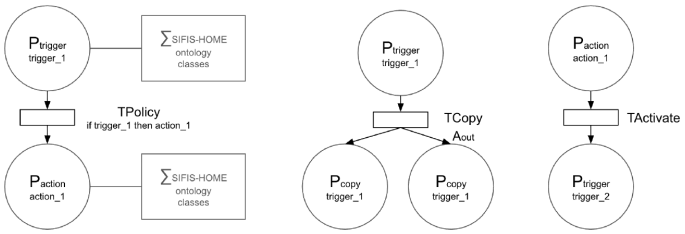


Figure 3: Semantic Coloured Petri Net used for conflict detection: policy transition (left), copy and activation transitions (right)

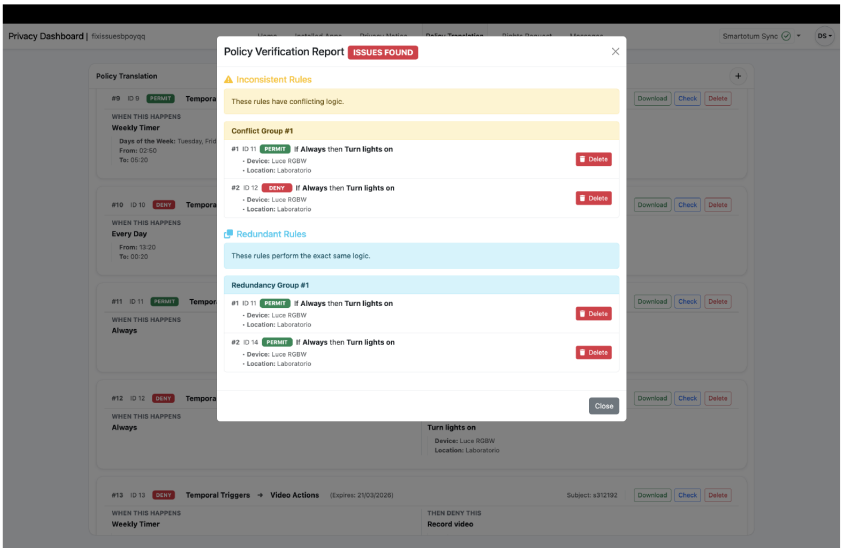


Figure 4: Policy verification report with inconsistent and redundant rules

6 Enforcement Pipeline and Integration with the Home-Automation Platform

6.1 Synchronisation

After login, the Data Subject triggers the synchronisation of homes and applications. For each home the back end performs four operations:

1. it retrieves the rooms, creating an additional room for devices without one;
2. it retrieves the devices of the supported types and assigns them to rooms;
3. it regenerates the local ontology from the synchronised devices;
4. it imports the rules already present in the `privacy_rule` topic of the DHT.

The synchronisation of applications imports their consent catalogue and the mapping between consents and device actions.

6.2 Propagation flow

When a consent is withdrawn or a rule is modified, the back end applies the following flow:

1. **Event.** The change is recorded in the database.
2. **Affected rules.** The services derive the privacy rules affected by the change.
3. **Translation and verification.** The PTP translates each rule into XACML and checks it for conflicts. A rule in conflict is kept for review by the responsible user and is not propagated.
4. **Publication.** The back end computes the device-level rule and publishes it to the DHT of the home as a `privacy_rule` topic.
5. **Acknowledgement.** The outcome of the publication is recorded in the audit trail.
6. **Notification.** The Data Subjects and the parties involved are notified.

When a consent is given, the corresponding automatic deny rule is first removed from the DHT, and only then is the consent marked as given. A rule that targets the whole home is stored once and published as one derived rule per compatible device.

6.3 Device-level rules

A rule published to the DHT names the target device type and device identifier, the daily time window, the days of the week and the expiration date:

```
{
  "topic_name": "privacy_rule",
  "value": {
    "target_topic": "domo_camera",
    "target_uuid": "<device uuid>",
    "time_start": "15:00",
    "time_end": "16:00",
    "days": ["Monday", "Wednesday", "Friday"],
    "expiration_date": "<yyyy/mm/dd>"
  }
}
```

On the home gateway, a privacy manager reads the active rules and marks the target device with the fields `privacy` and `privacy_until`. The user interface of the home then disables the actions on the device; for cameras, the gateway firewall also blocks the traffic towards the device (D2.2, D3.2).

6.4 Distributed state

The relational database of the dashboard is transactional, while the DHT is distributed and eventually consistent. The integration handles this difference in three ways:

- explicit synchronisation endpoints;
- timeouts and retries on the calls to the home-automation platform;
- recording the outcome of every publication in the audit trail.

The rules stored in the DHT remain the reference state for the devices.

7 Testing, Continuous Integration and Deployment

7.1 Testing

The back end is verified by an automated API test suite of **404 requests and 1,264 assertions**, organised in Postman collections and executed with Newman. A reduced collection of 161 requests and 317 assertions runs in the continuous integration pipeline. The tests cover all endpoint groups, the role-based access checks and the main flows, including:

- updates by the marketplace of metadata and consent catalogues;
- the snapshots kept in consent-withdrawal requests;
- the refresh of applications from the home-automation platform.

A *testing* mode replaces the home-automation platform with local mock data: systems, rooms, devices and `privacy_rule` topics. In this way the suite runs without depending on external services. The continuous integration enforces a minimum test coverage of 80%.

7.2 Continuous integration and delivery

A GitLab CI pipeline runs four stages on the integration branch:

1. **build:** builds the Docker image;
2. **test:** starts a temporary stack, waits for the health check and runs the API regression suite;
3. **publish:** publishes the image;
4. **deploy:** deploys it to the demonstration server.

7.3 Deployment

The application is packaged as a single image built in two stages: the front end is compiled with Node.js and then served together with the Flask back end. A Docker Compose file starts the application and a PostgreSQL container with persistent volumes. All sensitive settings are provided through environment variables:

- database credentials;
- token signing secrets;
- the endpoints of the home-automation platform;
- the marketplace integration token.

The same image therefore runs unchanged in development, testing and demonstration environments.

The system was evaluated in a laboratory deployment on a virtualisation server running the Docker stack, connected to a real instance of the home-automation platform.

8 Operational Metrics

Table 1 reports the benchmarks of the five operations most relevant to deployment, measured on the laboratory deployment. The consent-withdrawal benchmark used an in-memory proxy of the DHT, so its values describe the processing pipeline of the dashboard without the network latency of the home-automation platform. The benchmarks are reproducible with the scripts in `scripts/benchmarks`.

Table 1: Operational benchmarks (latencies in milliseconds; n = measured iterations after warm-up)

Operation	n	Median	Mean	p95	Max
API overall (59 endpoint groups)	384	14	177	908	1 828
Consent withdrawal (HTTP)	10	867	899	1 065	1 206
Consent withdrawal (end-to-end)	10	879	908	1 072	1 212
XACML translation (simple deny)	30	1 000	1 038	1 384	1 408
XACML translation (temporal deny)	30	1 002	1 046	1 423	1 462
XACML translation (3-subject batch)	30	3 040	3 188	3 558	3 609
SCPN analysis (10 rules)	50	0.26	0.28	0.41	0.47
SCPN analysis (25 rules)	50	0.60	0.70	0.91	2.07
SCPN analysis (50 rules)	50	1.58	2.41	2.26	37.06
OWL ontology loading	10	674	669	685	686

The measurements support five observations:

- **API responsiveness.** The median API response time of 14 ms shows that most endpoints respond interactively; the long tail is due to a few heavy operations such as the XACML translation.
- **Consent withdrawal.** A withdrawal completes in less than 1.1 s at the 95th percentile, including rule derivation, translation and publication.
- **XACML translation.** Translation takes about 1 s per subject and grows linearly with the number of subjects.
- **Conflict detection.** Detection stays below a millisecond up to 25 rules and below 2.5 ms on average for 50 rules, which is well within the needs of a household, estimated at 15–30 active rules.
- **Ontology loading.** Loading the ontology takes about 0.7 s and happens once, at start-up.

Table 2: Key metrics of the Data Privacy Dashboard (source: repository [5])

Metric	Value
REST API endpoints	74, in 14 blueprints

Automated API tests (Postman/Newman)	Full suite: 404 requests, 1,264 assertions; CI collection: 161 requests, 317 assertions
Minimum test coverage in CI	80%
Policy Translation Point code	about 2 000 lines of Python
Practical limit of SCPN analysis	about 50 rules (residential homes: 15–30 rules)

9 Installation and Use

9.1 Prerequisites

Running the dashboard requires Docker with Docker Compose. For development, Python 3.11 and Node.js are also needed.

9.2 Configuration

The configuration is read from a `.env` file in the repository root. The main groups of settings are:

- the run mode (`development`, `testing`, `demo`);
- the security secrets (`SECRET_KEY`, `JWT_SECRET_KEY`);
- the cookie settings;
- the PostgreSQL connection;
- the endpoints of the home-automation platform (base URL, DHT URL, JWKS URL and issuer) and the list of device types supported for rules;
- the marketplace integration token;
- the paths of ontologies and document templates;
- the allowed origins of the front end.

The secrets must be generated for each installation, for instance with `openssl rand -hex 32`.

9.3 Demonstration mode

In demonstration mode a single application container and a PostgreSQL container are started:

```
docker compose --env-file .env -f docker/docker-compose-demo.yml up -d --build
docker compose --env-file .env -f docker/docker-compose-demo.yml down -v
```

The application and the interactive API documentation (Swagger UI) are then reachable on port 5050 of the host.

9.4 Development mode

In development mode the database, the back end and the front end run separately:

1. start PostgreSQL (`docker compose up -d postgres`);
2. create a Python virtual environment and install `requirements.txt`;
3. apply the database migrations (`flask db upgrade`);
4. start the back end (`flask run --port 5050`);
5. install and start the front end (`npm ci`, `npm run dev`).

9.5 Testing mode and regression suite

With `FLASK_ENV=testing` the home-automation platform is replaced by the mock data in `backend/testing/data`, and Data Subject login does not contact the platform. The regression suite is executed on a temporary stack with the script `docker/scripts/test_image.sh`.

10 Conclusions

The Data Privacy Dashboard released with this deliverable brings the results of WP2 together in one open-source platform:

- **Roles.** It serves the three GDPR roles with dedicated views and binds Data Controllers and DPOs to the applications for which they are responsible.
- **Consents and rights.** It manages consents, rights requests and the documentation required by the GDPR.
- **Policies.** It translates user rules into XACML and verifies them for conflicts.
- **Enforcement.** It propagates consent decisions and rules to the devices of the home through the DHT of the home-automation platform.

The system is documented, tested by an automated API suite in continuous integration, deployed with Docker and measured on a laboratory deployment connected to a real home-automation platform. It is the policy front end of the ASSISTANTS pilot (WP3).

References

[1] L. Ardito, L. Barbato, P. Mori, A. Saracino. Preserving privacy in the globalized smart home: The SIFIS-Home project. *IEEE Security & Privacy*, 20(1):33–44, 2021. doi:10.1109/MSEC.2021.3118561.

[2] A. Monge Roffarello, L. De Russis. An Ontology-Based Approach for Setting Security Policies in Smart Homes. Springer, 2022.

[3] OASIS. eXtensible Access Control Markup Language (XACML) Version 3.0, 2013.

[4] F. Corno, L. De Russis, A. Monge Roffarello. Empowering End Users in Debugging Trigger-Action Rules. In *Proc. CHI '19*, ACM, 2019, pp. 1–13.

[5] ASSISTANTS Data Privacy Dashboard. https://github.com/Assistants-Project/Privacy-Dashboard (MIT licence).

Glossary

Acronym	Meaning
API	Application Programming Interface
CI	Continuous Integration
CNIL	Commission Nationale de l'Informatique et des Libertés
DHT	Distributed Hash Table
DPO	Data Protection Officer
GDPR	General Data Protection Regulation
JWKS	JSON Web Key Set
JWT	JSON Web Token
OWL	Web Ontology Language
PTP	Policy Translation Point
SCPN	Semantic Coloured Petri Net
XACML	eXtensible Access Control Markup Language