Rest Assured

SECURE DATA PROCESSING IN THE CLOUD

Deliverable D6.3
Revised Methodology and technical concept for Decentralized Data lifecycle Management
Release 1.0

The research leading to these results has received funding from the European Community’s H2020 research and innovation programme under grant agreement n° 731678.
<table>
<thead>
<tr>
<th><strong>Project Number</strong></th>
<th>731678</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Title</strong></td>
<td>RestAssured - Secure Data Processing in the Cloud</td>
</tr>
<tr>
<td><strong>Deliverable Type</strong></td>
<td>External Report</td>
</tr>
<tr>
<td><strong>Title of Deliverable</strong></td>
<td>Revised Methodology and technical concept for Decentralized Data lifecycle Management</td>
</tr>
<tr>
<td><strong>Nature of Deliverable</strong></td>
<td>External Report</td>
</tr>
<tr>
<td><strong>Dissemination Level</strong></td>
<td>Public</td>
</tr>
<tr>
<td><strong>Contractual Delivery Date</strong></td>
<td>31 October 2019</td>
</tr>
<tr>
<td><strong>Actual Delivery Date</strong></td>
<td>31 October 2019</td>
</tr>
<tr>
<td><strong>Contributing WPs</strong></td>
<td>All</td>
</tr>
<tr>
<td><strong>Editor(s)</strong></td>
<td>Dhouha Ayed</td>
</tr>
</tbody>
</table>
## Contributors

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dhouha Ayed, Cyrille Piatte</td>
<td>Thales</td>
<td>Data Lifecycle Security Methodology, Architecture and implementation of the Gatekeeper, Policies and Contracts specification, Data Protection Decision Point, Data Protection Enforcement Point, Introduction, Conclusion</td>
</tr>
<tr>
<td>Paul Mundt, Jodie Hatfield</td>
<td>Adaptant</td>
<td>Chapter 8: Application in RestAssured Use Cases</td>
</tr>
<tr>
<td>Nazila Gol Mohammadi, Ludger Goeke</td>
<td>UDE</td>
<td>Section 3 and Section 4</td>
</tr>
<tr>
<td>Mike Surridge</td>
<td>IT Innovation</td>
<td>Section 5</td>
</tr>
</tbody>
</table>
Contents

List of Figures ........................................................................................................ 6

1 Introduction ........................................................................................................ 8

2 Data Lifecycle Security: Methodology Overview .............................................. 10
  2.1 Data Lifecycle Security Roles ................................................................ 10
  2.2 Data Lifecycle Security Phases ................................................................ 10
  2.3 Context-Based Data Protection ................................................................. 11

3 Data Protection Policy Definition and Data Deployment ............................ 12
  3.1 Sticky Policy Definition and Data Storage Phase Overview ................. 12
  3.2 Data Protection Policy ............................................................................ 13
  3.3 Data Protection Contract and Consent Management ............................. 13
  3.4 Data Subjects’ Data Protection Preferences and Sticky Policy Definition 14

4 Data Protection Policy Specification Framework .......................................... 16
  4.1 Textual Patterns for Generation of Data Protection Policies ................. 16
     4.1.1 Part of Conceptual Model Relevant for Textual Patterns ............... 17
     4.1.2 Structure and Elements of Statements ........................................ 17
  4.2 Proof of Concept Implementation ............................................................. 20
     4.2.1 The Pay-As-You-Drive Use Case .................................................. 22
     4.2.2 The Pattern-Editor ..................................................................... 23
     4.2.3 Pattern-Instantiation-Editor ......................................................... 26
  4.3 Related Work ............................................................................................. 28
  4.4 Discussion ................................................................................................. 29

5 Compliance Checking and Change Management ......................................... 31
  5.1 Compliance Requirements ....................................................................... 31
  5.2 Change Management Requirements ....................................................... 31
  5.3 Change Management Approach ............................................................... 32
  5.4 Lawfulness of Processing ....................................................................... 33
  5.5 Other Regulatory Constraints .................................................................. 34

6 Communicating Data Protection Policies across Enclaves ........................ 37
  6.1 Secure Enclave ......................................................................................... 37
  6.2 Secure Data Processing Enforcement ..................................................... 37
6.3 Communicating Policy Changes Across Enclaves .......................................................... 37

7 Data Gatekeeper Implementation .................................................................................. 40
  7.1 Context Manager ........................................................................................................... 40
    7.1.1 Context Manager API ......................................................................................... 42
  7.2 Authentication Manager .............................................................................................. 42
  7.3 Data Protection Contract Manager ............................................................................. 44
    7.3.1 Data Protection Contract Manager API ............................................................ 44
  7.4 Sticky Policy Manager ................................................................................................ 45
    7.4.1 Sticky Policy Manager API ................................................................................. 46
  7.5 Data Protection Decision Point .................................................................................. 48
    7.5.1 Data Protection Decision Point API ................................................................. 48
  7.6 Data Protection Enforcement Point ............................................................................ 49

8 Application in RestAssured Use Cases .......................................................................... 50
  8.1 Data Lifecycle Security in the PAYD Use Case .......................................................... 50
    8.1.1 Overview and Background ................................................................................... 50
    8.1.2 Role of the Data GateKeeper .............................................................................. 50
    8.1.3 Integration with Other RestAssured Components ............................................. 50
  8.2 Data Lifecycle Security in the HPC Use Case ............................................................. 51
    8.2.1 Overview and Background .................................................................................. 51
    8.2.2 Role of the Data GateKeeper .............................................................................. 52
    8.2.3 Integration with Other RestAssured Components ............................................. 52

9 Conclusion ....................................................................................................................... 53

Bibliography ....................................................................................................................... 54
List of Figures

Figure 1.1: RestAssured Architectural Overview ......................................................... 8
Figure 2.1: Data Security Lifecycle ................................................................. 11
Figure 3.1: Sticky Policy Definition and Data Storage Phase Overview .............................................. 12
Figure 3.2: Overview of Data Protection Policy ................................................ 13
Figure 4.1: Simplified Conceptual Model .......................................................... 17
Figure 4.2: Elements of Data Protection Policy Statement ....................................... 18
Figure 4.3: Different types of statements ............................................................. 20
Figure 4.4: Examples of patterns ................................................................. 21
Figure 4.5: Excerpt of the 3PG-metamodel ........................................................ 21
Figure 4.6: Pattern-Editor tab for creation and management of data protection policy patterns ......... 25
Figure 4.7: EMF-tree editor of PAYD insurance service ........................................ 26
Figure 4.8: Instantiation of a data protection policy pattern in the Pattern-Instantiation-Editor .... 27
Figure 4.9: Instantiation of an obligation statement pattern in the Pattern-Instantiation-Editor ......... 28
Figure 6.1: Sticky Policy Partial Serialization ................................................ 38
Figure 6.2: Secure Data Processing Enforcement .............................................. 38
Figure 6.3: Enforcing Changes Across Enclaves ............................................... 39
Figure 7.1: General Architecture ................................................................. 40
Figure 7.2: Context Manager Operations ...................................................... 41
Figure 7.3: OpenID Connect protocol flow .................................................... 43
Figure 7.4: OpenID Connect in the RestAssured System ....................................... 44
Figure 7.5: Data Protection Contract Structure ............................................... 45
Figure 8.1: **PAYDv2 Architecture Overview** ............................................................. 51

Figure 8.2: **HPC Use Case Architecture Overview** .................................................. 52
1 Introduction

One of the main objectives of RestAssured is to provide solutions to specific technical concerns of data protection in the cloud, which are imposed by the dynamic, multi-stakeholder and decentralized nature of federated cloud system. This deliverable (D6.3) is the third deliverable of WP6 whose objective is to define a final methodology and implementation for managing a secure data lifecycle in a decentralized cloud setting based on the initial methodology that was defined in D6.1.

The goal of the deliverable is to define a methodology based on user consent and sticky policies that is used by the Data Gatekeeper. As described in D3.2, the Data Gatekeeper manages the data protection policies and the services governing the data life-cycle. It is responsible for deciding, based on the available policies and various constraints (legislation, context, location, etc.), which operations are allowed on which piece of data and to enforce user data protection policies in a decentralized computational environment.
- Section 8 illustrates the methodology as applied in the PAYD and HPC use cases;
- Section 9 concludes the document;
2 Data Lifecycle Security: Methodology Overview

In this section we describe the approach that the Data Gatekeeper uses to manage the data lifecycle security.

2.1 Data Lifecycle Security Roles

The security role definitions closely follow those defined by Article 4 of the GDPR[6]:

- The **Data Subject** is the natural person who is either directly or indirectly identifiable by the collected personal data.

- The **Data Controller** is the natural or legal person, public authority, agency or any other body (for example a trusted authority), which determines the purposes and means of the processing of personal data.

- The **Data Processor** is the legal person, public authority, agency or any other body which processes personal data, on behalf of the controller. In a cloud environment, this role is generally played by service providers.

- The **end user** is the person who needs to use the data by using a service provided by a service provider. He is the client of the service provider. He can generally be a data subject in the same time when he provides private data for the usage of the service.

2.2 Data Lifecycle Security Phases

Figure 2.1 illustrates the steps and roles that are required to successfully implement the data lifecycle security.

- Data classification and data subject preferences specification: Security has to be analysed at the creation of data. This analysis results in a set of security requirements and data protection preferences that are related to all the steps of the data lifecycle (storage, usage, etc.). The data subject shares these security requirements and preferences with the data controller, the entity that manages the data on behalf of the data subject. He also gives explicit consent about data he needs to be processed.

- Sticky policy definition and data protection plan: The data controller specifies or generates the sticky policies that are related to the data and plans the enforcement of these policies and the storage of the data according to these requirements and according to the regulation (see D6.2). He plans how and where to store the data (data structure, encrypted data base), where enforcement points are needed to be deployed, how to prove a processing consent, where and how to process data (enclave, homomorphic encryption, etc.).

- Once the data is stored, security checks are needed at rest and during data processing to ensure that security enforcement being used is effectively addressing the security policies. If not, data deployment and security enforcement must be revised in order to adopt changes and to address new risks.

- Revision and change: several types of changes can happen. For example, the data processing and storage context can change (data transfer, attack detection, system updates or degradation, location change, etc.). The data subject can also change his mind about his data protection preferences. All these types of changes require a revision of the data protection process and system and impacts the whole data lifecycle going from its classification to its storage and processing.
D6.1 focused on the sticky policy definition and data deployment phases. This document elaborates these phases (see Sections 3 and 4) and develops the data usage (see Section 6), compliance check and change management phases (see section 5), that are not covered by D6.1. The implementation of the whole process by the Data Gatekeeper is detailed in section 7.

### 2.3 Context-Based Data Protection

The shift of data storage and processing from a fixed desktop to dynamic environments, such as pervasive or cloud and mobile computing environments, the role of dynamically changing context has gained great importance for data protection access control and context-specific decision making, specially when end-users are mobile requiring seamless access to data and services.

Context is any information that can be used to characterize the situation of entities that are considered relevant to the inter-action between an end-user and a service, including the end-user and the service themselves. Context is typically the location, the execution environment of the service, state of people and physical objects.

A data controller needs to be context-aware by providing mechanisms to collect dynamic context information (execution environment, end-user properties and locations, etc.), to interpret them to detect relevant events in order to integrate them with data protection policy process and to manage an access to data at different granularity levels.

The RestAssured Data Gatekeeper solution relies on a Context-Based Access Control Model (CBAC) in order to support context representation and reasoning and to provide a data protection policy model that integrates context-specific decision making by taking into account the relevant contextual conditions. Our approach also uses an ontology-based approach to model and reason about contextual information. D6.1 details how context information are taken into account in data protection policies. This document implements this concept and integrates it in the change management process.
3 Data Protection Policy Definition and Data Deployment

This Section details the processes and assets provided by the Data Gatekeeper in the two first phases of the data lifecycle security (see Figure 2.1).

3.1 Sticky Policy Definition and Data Storage Phase Overview

A Data Controller makes usage of the Data Gatekeeper during this phase in order to play the intermediary role between the service provider/data processor and the DataSubject by allowing the service providers to specify the types of data processing they need to provide their services and the data subject to identify his data protection preferences.

A Data protection policy is a document provided by the service provider that informs about the collection and processing of the data that is needed by the service they provide. It takes into account the particularity of the service for data usage as well as the regulation (such as the GDPR).

A DataProtection Contract (see Figure 3.1) is agreed between the service provider and the data controller. It specifies the planned data processing needed by a service based on the Data protection policy after validating the data access and processing requirements of the service and verifying that they comply with the regulations.

The DataProtection contract is made available to the data subject in order to be used to manage the data subject consent. The data subject uses the Data Gatekeeper to get access to the Data Protection Contracts he needs before delivering his data to the data controller and can approve, disapprove data usage for each service or specify a context-based condition where he approves a usage.

Sticky policies are machine-readable policies that are logically linked to the data and that are automatically generated by the Data Gatekeeper based on the data subject protection requirements, his consents (based on data protection contracts) and legal obligations.

Figure 3.1: Sticky Policy Definition and Data Storage Phase Overview
3.2 Data Protection Policy

To comply with data protection legislation, e.g. GDPR, data protection policies (also named standard privacy policies in literature) are a widely used approach and an important legal foundation for data handling. Data controllers use data protection policies for informing their end-users (as data subjects) about the collection and processing of end-users data. These policies are provided by the data controller. According to the GDPR, service providers should provide a transparent data protection policy in a comprehensible way for end-users. In this deliverable (in Section 4), we also describe our approach for assisted generation of data protection policies using textual patterns. The proposed approach supports service providers (with the role of data controller) in their task of providing a comprehensible data protection policy as one of the data controller duties.

A Data Protection Policy can be used as input for generating a Data Protection Contract (see Figure 3.2). A Data Controller can also provide it as a document for informing the data subject about the data handling information. The Data Subject reads the provided data protection policy for getting awareness about data protection. More detail is described in Section 4.

![Figure 3.2: Overview of Data Protection Policy](https://restassuredh2020.eu//)

3.3 Data Protection Contract and Consent Management

A Service Provider or any entity that needs to process the data has to register with the Data Gatekeeper in order to specify the following information:

- the type of data that is needed by the service;
- the planned usage of data (actions and purposes).

This information is stored in a machine readable Data Protection Contract, that is signed before its storage in order to protect its integrity and made available for the Data Subject.

The Data Protection Contract represents a contract between a Service Provider that plans to use the Rest-Assured System in order to securely process personal data and the data controller (providing the DataGatekeeper).

The Data Protection Contracts contain 4 main parts:

- Service information in which the service provider delivers a summary description of the service that
includes the data protection policy document. It is made available for the data subjects in order to select the services they need to use.

- Configuration information, where the service provider specifies configuration and deployment information of its service in the Rest-Assured system, such as the database structure, the authentication system, cryptography system, usage of an SGX platform, etc..

- Data usage information. The service provider describes in this part the planned usages of personal data (statistics, marketing, the various functionalities of the service, public publish, etc.). For each usage, mandatory and optional data are specified.

- A signature that is issued in order to ensure integrity of the Data Protection Contract

The Data Protection Contract is a machine readable contract. The structure of the XSD file is described in D6.1, section 3.1.

3.4 Data Subjects’ Data Protection Preferences and Sticky Policy Definition

To overcome the difficulty in handling of data subjects’ data through different parties involved in a cloud computing environment, a possible solution is the use of sticky policies [14, 17], which have been proposed in various versions using different policy specification languages. Since the policy is transferred together with the data subjects’ data, every party knows how to handle the data. Sticky policies are data-oriented instead of service-oriented, allowing end-users to introduce their privacy requirements when using a service.

Although, when using sticky policies, the end-users are enabled to specify their data protection requirements and preferences, generation of sticky policies is not user-friendly and not usable for all end-users. These sticky policies are only considering data subjects data protection preferences without considering a specific service. On the one hand, end-users might tend to provide a rather strict sticky policy as they want their data to be protected. However, this restricts the service providers too much, even making service provisioning impossible for them. On the other hand, service providers are interested in getting the most benefit by using their data subjects’ data as freely as possible. Thus, they exploit the legal limits as far as possible in privacy policies with vague statements. This situation results in conflicts between the two parties.

RestAssured provides the Data Gatekeeper that negotiates the conflicts between the two parties (service provider with the role controller and the data subject). We use of the concept of sticky policies and uses them for the specification of data protection preferences of data subjects. This allows the data subject to adjust the data protection policy that is provided by the service provider. The negotiation of the interest conflicts between service provider and data subject enables the enforcement of the privacy preferences. In this way, RestAssured approach addresses the transparency as well as intervenability. Consequently, the data subjects are no longer obliged to consent to privacy policies that contain unnecessary processing of their data. This results in a better privacy protection in conformance to the GDPR (cf. D6.2).

The Data Gatekeeper focuses on capturing users’ data protection preferences and considering them in the sticky policies.

The Sticky Policy is generated for a Data Subject as soon as he/she registers to a specific service and applies his/her preferences regarding this service based on the data protection contract related to the service in the Data Gatekeeper.

A sticky policy can be separated in 3 main components:

- a reference part, that links the policy to the related pieces of data within the Rest-Assured system
- a content part, that specifies the Data Subjects requirement in terms of use and processing of personal data
• a signature part, that ensures the integrity of the Sticky Policy

The sticky policy is machine readable. The structure of the XSD file is described in D6.1, section 3.2.
4 Data Protection Policy Specification Framework

In D6.2, we presented our conceptual model for the specification of data protection policies. We also presented our pattern for creating a user-friendly, comprehensible visualisation of data protection policy statements. Based on our proposed pattern in D6.2, we enable developers to instantiate this pattern and provide a simple way for their end-users to express their data protection preferences and also to modify these statements. Furthermore, we provided data protection requirements based on the GDPR. In this deliverable, we provide textual patterns for generating data protection policies.

In this deliverable, we provide a Pattern-based data Protection Policy Generation (3PG)\(^1\). It assists service providers in generating data protection policies for their services. Our approach aims to satisfy the GDPR requirements. It is tool-supported and based on textual patterns. 3PG allows to instantiate textual patterns for creating a data protection policy for a specific service. The tool also allows to build or modify further patterns that can be used later in the instantiation process.

We present a conceptual model to describe which concepts play a role in the generation of data protection policies based on the GDPR. Although, the approach assists service providers, our patterns also consider end-users needs on data protection policies. The statements generated using our patterns have less complex constructs that enhance comprehensibility for end-users.

Please note that our proposed pattern (in D6.2) supports the interaction design for Data subject supporting data subjects in expressing their data protection preferences or requirements (cf. Figure 3.1). Here, in this section we present our approach with textual patterns to assist service providers in providing data protection policies for a specific service. The service providers use data protection policies as legal foundation for informing their end-users about data handling information.

The remainder of this subsection is structured as follows: Section 4.1 discusses our pattern-based data protection policy generation approach. It also describes how 3PG helps to create comprehensible and GDPR-compliant data protection policy statements in data protection policies. Section 4.2 presents the proof of concept implementation of our proposed approach along with a running example. We present related work in Section 4.3. Section 4.4 discusses our approach and gives hints about limitations and benefits and compares it to related work.

### 4.1 Textual Patterns for Generation of Data Protection Policies

Our approach of data protection policy patterns is to provide guidance on the specification of common types of data protection policy statements and their relevant obligation statements. The obligation statements are required by data protection regulations on particular actions performed by a service on end-users’ data. This approach decreases the creation time of data protection policies and makes them easier to draft. It also improves the quality of these statements in terms of comprehensibility to end-users and the completeness of information that should be made available to the end-users.

Our data protection policy patterns contain fixed text passages that mostly address either data protection requirements or data protection regulations, e.g. the GDPR. Additional to the fixed text passages, the structure of textual patterns may contain placeholders. These placeholders are replaced during the instantiation of a pattern by information regarding the actions on personal data of end-users performed by the considered service. In the structure of a data protection policy statement, the following information shall be provided: 1) actions performed on the end-user’s data, 2) the condition under which an action is performed on the user’s data, and 3) the purpose for which the user’s data is used.

We first present a simplified conceptual model of our approach, followed by a description of the necessary elements in the construct of statements that should be included in data protection policies.

\(^1\)This part of D6.3 is submitted for publication.
4.1.1 Part of Conceptual Model Relevant for Textual Patterns

Figure 4.1 shows the simplified conceptual model of our pattern-based approach. The data protection policy and consequently the service itself are restricted by data protection regulations, e.g., the GDPR. These regulations state the rights and duties with regard to the end-user’s privacy and data protection. The service provider is interested in adhering to the data protection regulations because substantial penalties are the consequence of disobeying the regulations. In addition to the data protection regulation, the data protection policy of a service should address data protection requirements as well. End-users use the service and provide their personal data to the service. A service is provided by a service provider and processes end-users’ data. In the context of our approach, a service provider represents a data controller as a legal entity providing a service which stores and/or processes personal data. The service provider has, with respect to the stored/processed end-users’ data, the obligations stipulated by the GDPR. The service provider that acts as data controller shall manage the data protection policies together with the data and data protection practices and other involved third-party service providers in a way that the services provided by these providers only perform actions that are permitted. Service providers provide also the data protection policies of the provided service. These data protection policies are used by the service providers as legal foundation for data handling information. The end-users read data protection policies for raising awareness about the provided service and how their data is handled by this service.

Within our approach, the end-user has the role of a data subject as defined in data protection regulation. Thus, an end-user represents a data subject who is a person whose data is stored and/or processed by the service. The end-user has, with respect to his/her personal data, the rights stipulated by the GDPR.

A data protection policy consists of one or several data protection policy statements. Service providers may use data protection policy patterns to generate data protection policies by instantiating these patterns. Each data protection policy statement is therefore an instance of a data protection policy pattern.

4.1.2 Structure and Elements of Statements

Figure 4.2 illustrates the different elements that a data protection policy statement may contain. The end-user’s data is processed by a service. These data is one of the categories that should be included in data protection policy statements as a key element to specify which data of the end-user is used by the service. Thus, each statement relates to the data provided by the end-user. The statement expresses conditions...
and rules that refer to the actions that are performed by the service. The associations of end-user’s data, action, and purpose are important because it is necessary to track which actions are allowed to be performed for which purpose. The service performs actions on end-users’ data. These actions are part of the data protection policy statements that will be specified using the data protection policy patterns. The different categories of actions define what can be done with the end-users’ data that is provided to the service. Categories of actions are as following:

- Collect: Collection of end-users' data that is directly or indirectly supplied by the end-user
- Store: Storing of data supplied by the end-user
- Process: Processing of data resulting in new data
- Transfer: Transferring data to other locations
- Share/Disclose: Sharing data with other parties (is a kind of transfer action)
- Migrate: Migrating is a kind of transfer action that changes the location of data. This action has been considered separately because of the special requirement of the GDPR on migration.
- Notify: Notification of the end-user of specific events

Each of these actions can be included in the data protection policy statement as one of its elements.

Every action has a purpose. The purpose of an action is another category of information that is part of a data protection policy statement (see Figure 4.2). We distinguish four categories of purposes. These categories of purposes are as following:
• Service provisioning as purpose defines whether the action is necessary for service provision. It means that the action is necessary for providing the service, e.g. account information for online banking.

• Marketing as purpose defines whether the data supplied by the end-user is used for marketing.

• Analytics as purpose defines whether the end-users’ data is used for performing analytics. Analytics addresses the usage of end-users’ data for the creation of statistics.

• Advertising as purpose defines whether the end-users’ data is used or shared with others for advertising.

Because of this purpose-binding of actions, it is possible to decide whether the usage of the end-user’s data is actually necessary to provide the service or whether the end-user can decide whether to use this service. Marketing, analytics, and advertising are optional purposes, stating that the action is not actually necessary for the service to function. The difference between marketing and analytics is as follows. Marketing means the usage of the end-user’s data to inform new end-users about the service, whereas, advertising means the usage of the data to provide advertisements to the end-user that provides the data.

Using the data protection policy, service providers can communicate their data protection practices to the end-users. The generated data protection policy contains in addition to the data protection policy statements further statements for applied obligations. These obligation statements can, for example, specify the prohibitions in the processing (action) of the data for specific purposes or retention/deletion conditions (see Figure 4.3).

We provide textual patterns for generic data protection policy statements as well as more specific data protection policy statements. Examples of specific data protection policy statements (see Figure 4.3) are as following:

• A collection statement specifies which data is allowed to be collected by the service provider. This means, it will restrict the collect action of the service provider.

• A usage statement specifies for which purpose the end-users’ data is allowed to be used and processed.

• A storage statement specifies how long and where the data will be stored.

• A share/disclose statement defines the rules and conditions specifying which data of the end-users is allowed to be disclosed to which audience.

• A migration statement describes the rules and conditions for the migration of the data to different locations.

• A notification statement specifies the rules and conditions regarding the question about what the end-user needs to be informed and how.

• A consent statement describes in which cases and under which condition the end-users are required to give their consent.

Depending on a specific data protection policy statement with its concerned action types, one or more obligations can be relevant to the statement. For example, upon a storage statement the obligation statement should be added to specify when and how the service provider deletes this data. In our work, we provide also patterns for obligation statements that can be related to data protection policy patterns. The structure of obligation statement patterns is similar to the structure of data protection policy patterns.

Both types of patterns include the following constructs:

• Fixed text: Text passages of a pattern that cannot be modified during the instantiation of the pattern.
• Generic placeholders: References to certain types of elements of the considered service. During the instantiation of a pattern a generic placeholder is replaced by information of a service element, whose type corresponds to the referenced type(s) in the generic placeholder. Generic placeholders are marked by opening and closing square brackets ([ ]). As an example, the generic placeholder within square brackets in the generic data protection policy pattern in Figure 4.4 (see bullet point 1.) references all types of the category action. During the instantiation of the data protection policy pattern, the designer of a specific service can replace the placeholder by the information of a service element of a subtype of action. The Figure 4.4 also represents an example for a collection statement pattern instance (see bullet point 3.) that has been instantiated based on the corresponding pattern (see bullet point 2.).

For structuring of both fixed and appropriate generic parts of our patterns, we have identified a number of keywords for each of the elements, e.g. for expressing actions or conditions such as time constraints. We have analyzed the state of the art and terminologies used in current data protection policies. Then, we created a catalog of data protection requirements based on the GDPR [D6.2], we keep us in our textual patterns with the terms and keywords used there. For example, multiple terms may be used in order to express “collecting” end-users’ data, e.g. obtain, gather, get, receive, etc. In our patterns, we use our identified keywords (in this case “collect”), which are based on the GDPR terminology. In this way, the data protection policies created using our textual patterns will be consistent and uniform.

Further examples for data protection policy and obligation statement patterns are presented in the next section.

4.2 Proof of Concept Implementation

We provide tool support for our textual data protection policy patterns and obligation statement patterns, namely Pattern-based data Protection Policy Generation (3PG)-tool. This tool provides two following

2. A **collection statement pattern**: The [Service] will [Collect] [End-User’s Data] for [Service Provisioning].

3. An **instance of collection statement pattern**: The navigation service will collect your location data for service provisioning.

**Figure 4.4: Examples of patterns**

Both editors are realized as Eclipse plugins by using the **Eclipse Modeling Framework (EMF)** (see Figure 4.5). Eclipse plugins are implemented in the programming language JAVA. EMF supports the implementation of Eclipse plugins by providing a model-driven approach. Therefore, the framework provides a set of language constructs for the creation of metamodels that enable the specification of an abstract syntax of model information. The provided constructs correspond to the elements of class models that are part of the **Unified Modeling Language (UML)**. Thus, EMF-metamodels contain elements like classes and associations. Figure 4.5 shows an excerpt of the EMF-metamodel that defines the abstract syntax of our data protection policy and obligation statement patterns and their instances. This excerpt focuses on the specification of patterns. As it can be seen, the different elements of the patterns are defined by metamodel classes. Relations between the pattern elements are specified by associations. A more detailed discussion of the EMF-metamodel in Figure 4.5 is given in Section 4.2.2.

**Figure 4.5: Excerpt of the 3PG-metamodel**
The Pattern-Editor and Pattern-Instantiation-Editor each implement a domain layer for the representation of pattern or instance information, respectively, at runtime of the editors. In the context of Eclipse plugins, this information is represented in the form of appropriate Java objects. The Java classes that specify these Java objects are generated using EMF. It additionally generated the source code of a serializer, and deserializer that enable the storing of pattern and instance information in a corresponding XML format.

The editor for creating the information of a service is implemented by using EMF too. Thus, the abstract syntax of the service information is defined by a separate EMF-metamodel. This metamodel not only defines information about services itself, but also specifies information for data protection and data protection regarding a service. The specified data protection and data protection information corresponds to the definitions in Figure 4.2. For the domain layer of the service information editor, the appropriate Java classes are generated by using EMF. These Java classes specify the Java objects for the representation of service information during runtime of the editor. The graphical user interface of the editor is realized as a standard EMF-tree editor. This tree editor is generated completely by EMF. It enables the representation of service information in the structure that is defined by the metamodel. The functionality of the service information editor includes the serialization and deserialization of service information in an appropriate XML format. A more detailed description of the service information editor is given in Section 4.2.3.

The Pattern-Editor and Pattern-Instantiation-Editor are explained in the Sections 4.2.2 and 4.2.3. The description of the editors is based on the Pay-As-You-Drive (PAYD) Insurance use case. The PAYD insurance service is presented in Section 4.2.1.

### 4.2.1 The Pay-As-You-Drive Use Case

This section presents an example from PAYD use case.

The PAYD use case is a cloud computing service for automotive insurance companies. PAYD enables insurers to offer innovative, cost-effective and usage-based automotive insurance products. This is achieved by collecting and analyzing the driving data of insurance customers. Since this driving data is personal data of the insurance customers, an appropriate data protection and data security have to be realized.

The GDPR is one of the effective regulations for the provided services. Accordingly, insurers take the data controller role from GDPR. The insurance customers as the end-users (the drivers) have the role of data subject from GDPR.

The first service that is provided is the PAYD web application. This web application is used by the insurance customers (end-users). By using the PAYD web application, the insurance customers are able to register for an insurance product and get information about their current insurance conditions. The end-users provide their personal data as insurance customer data to the PAYD web application. The access, processing and storage of the insurance customer data is described in the provided data protection policy.

The second service involved in this scenario is a telematic service. This service is responsible for receiving the driving data of the insurance customers. The collected driving data is transferred to the cloud infrastructure of the corresponding insurer. There, the driving data is analyzed by insurance analysts for determining the individual insurance premiums for insurance customers.

The data protection policy should include information on the access, processing and restrictions provided by the service providers with regard to the amount of the driving data.

In the description of the Pattern-Editor (Section 4.2.2) and Pattern-Instantiation-Editor (Section 4.2.3), the processing of personal customer data and driving data within the PAYD insurance scenario is considered as a running example. Since the discussion focuses on the design of patterns and their instances, only partial data from the PAYD insurance scenario is considered.
4.2.2 The Pattern-Editor

The Pattern-Editor of the 3PG-tool allows designers to create and manage data protection policy patterns and obligation statement patterns. For this purpose, the 3PG-tool provides an editor tab for each type of pattern. In the following, we describe the main constructs of a pattern.

Figure 4.5 shows a part of the metamodel that defines the abstract syntax of our patterns. The basic properties for data protection policy and obligation statement patterns are defined by the metaclass Pattern. A pattern, which is identified by an unique identifier, consists of fixed text and generic text. The fixed text represents the text in a pattern that cannot be changed during the instantiation of the pattern. In contrast to the fixed text part of a pattern, different types of generic text are replaced by appropriate information during the pattern instantiation. A generic text has a length and is inserted at a certain position in the fixed text (marked with placeholders). In the following, the different types of generic text are described.

Patterns are instantiated in a semi-automatic manner for a targeted service definition whose information is represented by service elements of different types. A service element placeholder defines a placeholder that references one or several types of service elements. During the instantiation of a pattern, a service element placeholder is replaced by the information represented by a service element of the appropriate type (see Section 4.2.3). Such a placeholder in a pattern is indicated by an opening and closing square bracket.

Another type of generic text is a text choice. A text choice contains one or more text choice elements. During the instantiation of a pattern, one of the specified text choice elements can be selected and replaces the text choice placeholder. A text choice element represents either an invariable keyword or a piece of text that contains a placeholder that should be replaced by appropriate information. This information is supplied by the designer of the service, who instantiates the pattern. Such a text choice placeholder is indicated by the symbol “*” (see Figure 4.9 at the bottom).

Free text is the last type of generic text. It can be replaced by any appropriate information by the designer during the pattern instantiation. It contains only a description that explains the domain of information that the designer should specify. Free text is indicated by angle brackets.

Figure 4.6 shows the definition of a data protection policy pattern for services in the dedicated tab of the Pattern-Editor. The middle part in the editor tab enables editing of data protection policy patterns. It contains a list named Types of Service Information. This list includes all types of service element instances that can be referenced by a service element placeholder. The list is created, when the appropriate XML-file, whose format corresponds to the 3PG-metamodel (cf. Section 4.2), is loaded into the Pattern-Editor for the first time. The necessary information in the list is based on the names of the non-abstract metaclasses of the EMF-metamodel that specifies the service information. Only the names of metaclasses representing certain types of actions are replaced by appropriate verbs to increase the readability of the patterns. As an example, for the metaclass “ProcessUse” the name “process” is included into the list. For the actual editing of a data protection policy pattern, a stylized text field is provided, which is placed below the text field for depicting the pattern ID. A service information type can be inserted into a data protection policy pattern by double-clicking on the corresponding list item, or can be manually typed into the stylized text field. In the case of manual entry by designers, a syntax check ensures that incorrect type names and other syntax errors are detected. Figure 4.6 shows the definition of a data protection policy pattern that refers to the action of storage that is performed in the context of a service. The storage action concerns geodata that belongs to end-users of the service for service provisioning.

The table below the stylized text field for editing data protection policy patterns contains the obligation statement patterns that are related to the current data protection policy pattern. The relations of a data protection policy pattern to obligation statement patterns can be added and removed through a dialog that is started by clicking the “Edit” button.

On the left-hand side of the editor tab, the catalog of data protection policy patterns is shown. It provides functionalities for starting the development or modification of data protection policy patterns, as well as the deletion of data protection patterns.
The right-hand part of the editor tab displays the specified obligation statement patterns. These obligation statement patterns have been defined using the second editor tab of the Pattern-Editor. In contrast to data protection policy patterns, the obligation statement patterns specify the additional information that indicates whether an obligation is stipulated by the GDPR. Since the functionalities and techniques for developing and managing obligation statement patterns are quite similar to corresponding features for data protection policy patterns, this editor tab is not further explained.
Figure 4.6: Pattern-Editor tab for creation and management of data protection policy patterns
4.2.3 Pattern-Instantiation-Editor

The Pattern-Instantiation-Editor allows the instantiation of existing data protection policy patterns and their corresponding obligation statement patterns, which are defined with the Pattern-Editor. During the instantiation, the generic text of a pattern is replaced by concrete information.

In the case of generic text as a service element placeholder, the placeholder is replaced by elements from the service to which the data protection policies and obligation statements should apply. In the Pattern-Instantiation-Editor, these service elements are represented by service element instances. The name of a service element instance, which is identified by a unique identifier, represents the information of a service element of a particular type. The type of the service element is also a property of the service element instances. The service element instances for all elements of the targeted service are created the first time the XML-file that includes the pattern and instance information is loaded in the Pattern-Instantiation-Editor. To do so, the appropriate XML-file with the service information is also loaded for provision of the service information that is required to create the service element instances.

In our example, service element placeholders are replaced by service element instances that represent service elements from the PAYD use case (cf. Section 4.2.1). Figure 4.7 shows the definition of the service elements of the PAYD use case in the corresponding EMF-tree editor. The upper tree view lists the different service elements. A service element comprises the actual service information and its type. The service information displayed in the tree editor, is contained in the service element instances created by the Pattern-Instantiation-Editor. During the instantiation of data protection policy and obligation statement patterns for the PAYD use case, service element placeholders are replaced by service element instances of the appropriate type. For clarity, only a partial extract of the PAYD service information is represented in the tree editor.

In the following, the service information in the tree (see Figure 4.7) is briefly explained. The service provider is an “insurance company”. Regarding the service, data subjects are represented by the “insurance customers”. The service performs actions to process and store geodata in the form of its customers “driving data” to determine the insurance premium for each customer. Accordingly, the purpose for storing and processing driving data is the “provision of the insurance service”. Additionally, it processes and stores “personal customer data” as profile data for service provisioning.

The properties of the currently selected service element are displayed in the lower property panel of the tree editor. These properties include the name of the selected service element and its unique identifier (Uuid) as well as associations of the service element with other service elements.

Figure 4.7 shows the properties for the service element “insurance customer” of the type data subject. It contains the reference Provided Data that implements the association of a data subject to the data that is provided to the relevant service. Within the PAYD use case, insurance customers provide their driving data...
Figure 4.8 shows the instantiation of the data protection policy pattern “Ppp_001” in the Pattern-Instantiation-Editor. Before this instantiation is discussed in detail, the general structure of the Pattern-Instantiation-Editor is explained in the following.

Data protection policy patterns and their corresponding instances are assigned to the types of actions in the service under consideration. For the PAYD use case, the appropriate data protection policy patterns are assigned to the actions store and process (see Figure 4.7). This assignment is indicated by a tree in the left-hand side of the editor (see Figure 4.8). The tree for the PAYD use case in Figure 4.8 shows that only such data protection policy patterns are assigned to a certain action, when a data protection policy pattern contains a service element placeholder that refers to the type of the action. For example, all data protection policy patterns that refer to the process action contain the service element placeholder “[process]”.

A data protection policy pattern for a particular action can be instantiated by selecting the pattern in the tree and clicking the Instantiate-button (shown in Figure 4.8). The created instance can then be edited in the middle panel of the editor. In the following, the different components of editing are explained in a top-down order. First, the editor panel provides a text field for editing the ID of the data protection policy instance. Another text field displays the related data protection policy pattern. The table (in the middle of the editor panel) allows replacing the generic text with information of the service under consideration. To do so, the table lists all definitions of generic text in the associated pattern in the Definition column. The Mapped Value column contains the information that replaces the corresponding placeholder in the instance. In the instantiation of the data protection policy pattern Ppp_001 (see left-hand side of Figure 4.8), the Definition column contains only service element placeholders. Thus, the Mapped Value column contains only service element instances. In this example shown in Figure 4.8, the service element placeholders “Service”, “store” and “GeoData” have already been replaced by appropriate service element instances. The value for the placeholder “store” that represents the action is assigned automatically. The replacement of a generic text is started by a double-click on the appropriate row in the table. If a substitution references a service element placeholder, the service element instances that are suitable for substitution are displayed in a table on the right-hand side of the editor. A service element instance for the substitution is selected by double-clicking on the appropriate row. Figure 4.8 shows the substitution for the placeholder “[DataSubject]”. In the right-hand part of the editor, the table provides the only relevant service element instance, the “insurance customer”, as a data subject.

At the bottom in the middle of the editor, the text field displays the current text of the data protection policy instance during the instantiation process. If a generic text is not replaced yet, the generic text of the related pattern is displayed.

After the instantiation of a data protection policy pattern, the relevant obligation statement patterns are displayed in the tree view under the created data protection policy instance. In Figure 4.8 (left-hand side), which corresponds to the data protection policy pattern Ppp_001 (gray colored), that is related to the data protection policy instance PppInst_001, the obligation statement patterns Osp_003 and Osp_004 are associated to this instance.

Figure 4.8: Instantiation of a data protection policy pattern in the Pattern-Instantiation-Editor
Figure 4.9: Instantiation of an obligation statement pattern in the Pattern-Instantiation-Editor

The instantiation of obligation statement patterns is similar to the previously described instantiation of data protection policy patterns. Figure 4.9 illustrates the instantiation of the obligation statement pattern Osp_003. Here, the replacement of a generic text in the form of a variable text choice containing a wildcard is shown. The Pattern-Instance-Editor provides an appropriate dialog (see Figure 4.9) for the substitution of the text choices.

4.3 Related Work

In software engineering, patterns are an adequate solution approach for solving problems that occur frequently in a specific domain. There are various types of patterns as for example design patterns, requirements patterns and risk patterns [4, 3]. In this work, we focus on pattern-based data protection policy statements. To provide an appropriate guideline for designers in establishing a data protection policy document, textual patterns can be used.

Standard Privacy Policies. A variety of research approaches address deficiencies of privacy policies. Pollmann and Kipker [16] suggest a competitive rating system, where end-users and experts can rate service providers in their attitude toward personal data processing [16]. This approach encourages the service providers in respecting the end-users’ privacy. There are several works that address improving comprehension of the privacy policies by redesigning the structure of privacy policies. Kelly et al. motivate their new privacy policy layouts by the problem of long full-text documents [11]. The authors discover that lengthy full-text policies causing disadvantages in comprehending and discovering required information by end-users. They analyze how well end-users were aided in their explorations of privacy policies of websites with the Expandable Grid of the Platform for Privacy Preferences (P3P) designed by the CyLab Usable Privacy and Security Lab. User experiments by Kelley et al. shows that users are confused with complex structure of the Expandable Grid. Therefore, Kelley et al. designed another layout and different arrangement of information. But this new layout came with the serious disadvantage of information loss. As a result, Kelley et al. decided to “bring back more of the detailed information that privacy policies can provide without overwhelming users” [11]. They adjusted the statements to fit them into a two-dimensional grid layout, the Simplified Grid that was later transformed into the Privacy Nutrition Label. A study confirmed that the presentation and comprehension of information within the Privacy Nutrition Label is less time-consuming and less complex than the scanning of lengthy full-text policies [11] [12]. Ghazinour et al. [10] develop a Model for Privacy Policy Visualization that facilitates understanding privacy policy statements. They address the problem of end-users who are not paying careful attention to privacy policies when using a web service. In addition, the authors provide a tool for the service providers to improve, debug and optimize their privacy policies [10]. In another paper, Ghazinour et al. [9] address the privacy requirements of purpose, visibility, granularity, retention, and constraint using their Model for Privacy Policy Visualization.
It defines different elements (e.g. entity, relation, privacy notation, group attributes, and default values) that are displayed by various symbols. Our work differs from this kind of visual representation approaches, since we aim at improving the quality of data protection policy documents using textual patterns. There is also an obligation from the regulation that the textual versions of data protection policies should be improved.

Bhatia et al. [2] analyse and identify semantic roles in the privacy practice statements provided by service providers [2]. We considered their analysis and results in the structure of our textual patterns for data protection policy statements.

**Textual Patterns in Requirements Engineering.** Textual patterns are used in specification of requirements [19]. Withall [19] identifies 37 requirement patterns [19]. All of the 37 requirement patterns are divided into eight domains: fundamental requirement patterns, information requirement patterns, data entity requirement patterns, user function requirement patterns, performance requirement patterns, flexibility requirement patterns, access control requirement patterns, and commercial requirement patterns. Withall provides guidelines and examples for formulating requirements in natural language. He aims at writing textual requirements, which also consider domain knowledge. Our work differs from Withall’s, because we provide patterns for data protection policy statements based on GDPR.

Pohl [15] introduces patterns which describe scenarios in a structured textual format. Pohl’s requirements templates [15] include possible exception scenarios as well. In agile software development, for specifying requirements there are textual patterns for drafting the user stories [5]. The idea of using a structured textual format in our data protection policy statements pattern is similar to these requirements templates. These works do not provide specific notations to document data protection requirements nor any other non-functional requirements, and only consider functional requirements.


Beckers et al. [1] present a method for specification of security requirements for cloud computing systems. The authors provide textual security requirement patterns. We build our approach upon this work and present more specific patterns for data protection policy statements and obligations. For this purpose, our work extends the approach presented by Beckers et al. [1] with adding elements that address data protection requirements and data protection requirements. The added elements conform to the GDPR.

### 4.4 Discussion

The procedure presented in this deliverable was developed based on discussions with practitioners from RestAssured project partners. Parts of our approach have been discussed with data protection consultants. The consultants mentioned that this structured approach supports the specification of data protection policies and is even useful for data protection requirements.

It increases the usage of models of the service and generating data protection policies in a semi-automatic manner instead of texts from standards and law, which eases the effort of understanding the service and its data handling process and data protection practices. Additionally, it facilitates the comprehension of the data protection policy document and unifies used terminologies in documentation significantly. Furthermore, our approach provides the means for abstraction of a complex system and structured reasoning for data protection. It also constrains that the data protection policy statements describe concretely and explicitly which actions are performed for which purpose on the end-users’ data. Accordingly, instead of long, complex and abstract statements, our approach leads to simple, short and more comprehensible statements for end-users. Furthermore, data protection policies are uniform and consistent through the adoption of a set of pre-defined, GDPR-compliant terms and keywords for describing data protection matters.

One issue that needs further investigation is scalability, both in terms of the effort needed by the require-
ments engineers and service designers in order to enter all information about the service. We will use the approach for different scenarios to investigate if the method scales for different domains and applications, as well.

We aim to conduct an empirical study with our tool in order to analyse the amount of time that can be saved when using our approach. Furthermore, to analyze the achieved user experience by the end-users when reading (in terms of comprehensibility) such documents generated using our approach. We aim to compare it against conventional text based approaches. Our approach will also undergo a series of further usability tests, which shall discover issues with its use in a productive environment. We aim to identify usability issues and resolve these in order to further improve the user experience (for designers of the service).
5 Compliance Checking and Change Management

5.1 Compliance Requirements

As outlined in Sections 1 and 2, the Sticky Policy framework is designed mainly to regulate access to personal data. It supports acquisition and encoding of a data subject’s consent for their data to be used for specific purposes. Each purpose is associated with a service function defined in the data protection policy. The data subject then uses the Data Gatekeeper to grant (or withhold) consent allowing their data to be used by each service function. The Data Gatekeeper creates a sticky policy associated with their data which encodes the access rights granted by their consent decisions. The Data Gatekeeper then provides a policy decision point, which is used by policy enforcement points on the path between a service and the data to determine which data can be used, and obtain keys embedded in authorization tokens giving access to the data. The process using the data can also be executed in a secure enclave, allowing the data to be sent in encrypted form and only decrypted in the enclave, preventing other access even by privileged users of the underlying cloud infrastructure.

The GDPR does not say that personal data may only be processed by consent of the data subject. In Article 6, it defines circumstances under which processing is permitted without consent, e.g. one may process personal data to protect the vital interests of a natural person (who need not be the data subject). In Articles 8 and 9, it defines restrictions on processing even if the data subject does consent, unless additional safeguards are provided, e.g. Article 8 says that processing of data from children requires the service operator to seek approval from a parent or guardian, while Article 9 says that processing special categories of data by consent is only lawful if the consent is informed consent, in which the data subject is fully informed about potential risks. Article 9(4) also says that for three of the special categories (biometric data, genetic data and health data), national regulators may impose additional restrictions these may specify that processing is not permitted under any circumstances or impose other requirements on how the data is handled or protected. Since these regulations are not harmonised across the EU, national regulators may also impose restrictions on the free movement of data covered by Article 9(4).

Compliance checking for personal data processing therefore covers the following:

• whether data is processed under one of the situations where this is allowed by the GDPR, including but not limited to processing by consent;

• whether additional requirements are met when processing special categories of data or data from children; and

• whether national regulations are met in the case of biometric, genetic or health data.

For impersonal data, the approach taken by RestAssured is to assume processing is still by consent, but as there is no data subject, consent is given by the data owner, i.e. the stakeholder that created the data or sent it to be processed in the cloud. The other stipulations of the GDPR do not apply in this case, so there are no exceptions under which consent is either insufficient or not required. In some sectors, processing may be subject to other regulations, e.g. regarding classified or financial data, but these are beyond the scope of the RestAssured project.

5.2 Change Management Requirements

Cloud based services are by their nature dynamic systems, in which various changes can occur:

• in the preferences and restrictions imposed by stakeholders including the data subject;
in the distribution of data and processing between data centres and between devices in a data centre;

• in the security mechanisms that are available to and/or actively used by each service; and

• in regulatory constraints on the storage, transfer and use of data.

The GDPR Article 25 requires data controllers to ensure data is protected by organisational and technical means, taking account of the nature and purpose of processing and the risks of various likelihood and severity for rights and freedoms of natural persons. This must be done both at the time of the determination of the means for processing and at the time of the processing itself, so security and risk levels must be taken into account in both the system design and when managing dynamic changes in the configuration or operation of such a system.

5.3 Change Management Approach

To address these requirements, RestAssured WP7 has developed a methodology for risk assessment that incorporates:

• design-time risk analysis and mitigation planning by service developers;

• deployment-time analysis of the context for a specific deployment, including analysis of stakeholder impacts and regulatory requirements;

• continuous automated risk analysis of dynamic changes incorporated into the autonomic management of cloud services.

This methodology is described in Deliverable D7.3 [18]. Using this approach, compliance can be addressed at different points in the lifecycle of a system or application. In particular:

• the legal basis for processing under the GDPR is normally determined at design time, and SSM models used to find technical measures that address GDPR requirements;

• regulatory requirements arising in a specific deployment are detected in the deployment-time analysis using CSAP models to analyze stakeholders and regulatory context;

• the adequacy of security measures to address risks is determined using an SSM model based on the initial design time analysis, but incorporating deployment specifics;

• subsequent observed or proposed changes are analyzed using updated versions of this SSM model, providing alerts of any non-compliant changes or elevated security risk levels.

The use of both static (design/deployment time) and dynamic (run time) risk assessment, combined with autonomic adaptation to activate security measures where needed, or to shut down personal data processing if this cannot be done, ensures system does comply with the GDPR Article 25. These aspects are covered in detail in D7.3. Here, we focus on change management with respect to other regulatory compliance requirements: the legal basis for processing, and the regulatory context that may constrain the storage, transfer and use of data.
5.4 Lawfulness of Processing

Under the GDPR Article 6, processing is lawful only if and to the extent that at least one of the following applies:

a) the data subject has given consent to the processing of his or her personal data for one or more specific purposes;

b) processing is necessary for the performance of a contract to which the data subject is party or in order to take steps at the request of the data subject prior to entering into a contract;

c) processing is necessary for compliance with a legal obligation to which the controller is subject;

d) processing is necessary in order to protect the vital interests of the data subject or of another natural person;

e) processing is necessary for the performance of a task carried out in the public interest or in the exercise of official authority vested in the controller;

f) processing is necessary for the purposes of the legitimate interests pursued by the controller or by a third party, except where such interests are overridden by the interests or fundamental rights and freedoms of the data subject which require protection of personal data, in particular where the data subject is a child.

The functions of the system are determined at design time, so the decision as to which of these will be used for each function is also determined at design time. If the SSM tool from WP7 is used with the RestAssured knowledge base, a regulatory compliance threat will be generated for each process using personal data. For each possible legal basis there is a control strategy that sometimes includes technical measures to be incorporated into the design and implementation of system assets, or applied in the deployed system.

Although the legal basis used for personal data processing can be determined (statically) at design time, some of the consequent rights and restrictions are affected by dynamic changes, as explained in Table 5.1.

<table>
<thead>
<tr>
<th>GDPR Article</th>
<th>Legal Basis</th>
<th>Dynamic Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1(a)</td>
<td>Consent</td>
<td>Decisions by the data subject to grant or withhold consent.</td>
</tr>
<tr>
<td>6.1(b)</td>
<td>Contract</td>
<td>None, following the decision by the data subject to engage.</td>
</tr>
<tr>
<td>6.1(c)</td>
<td>Legal obligation</td>
<td>None, unless laws change imposing different legal obligations on the data controller (which we assume to be slow enough to be handled via system redesign).</td>
</tr>
<tr>
<td>6.1(d)</td>
<td>Vital interests</td>
<td>Recognition that vital interests of a natural person are at risk.</td>
</tr>
<tr>
<td>6.1(e)</td>
<td>Public interest</td>
<td>None, unless laws change alter the understanding of what constitutes a public interest.</td>
</tr>
<tr>
<td>6.1(f)</td>
<td>Legitimate interests</td>
<td>Recognition that the data subject’s rights and freedoms override the legitimate interest, e.g. because the data subject is a child.</td>
</tr>
</tbody>
</table>

Table 5.1: Dynamic changes related to the legal basis for processing

The technical measures proposed by RestAssured are incorporated 'by design', but are able to handle dynamic changes that affect whether data processing should be permitted:

- Consent: should be implemented by using RestAssured Sticky Policies, allowing consent decisions by the data subject to be captured, and the consequential access restrictions determined and enforced;
• Vital interests: requires a policy exception allowing consent restrictions to be bypassed if vital interests are found to be at risk, also known as a 'break-the-glass' feature;

• Legitimate interests: requires a means to determine whether these interests are overridden by interests or fundamental rights of the subject, based on criteria such as the age of the data subject.

All these measures are included in the RestAssured domain model from WP7 and so should appear in a design time risk model created using the SSM tool. Only the first (use of Sticky Policies to implement processing by consent) was a focus in the RestAssured validation scenarios. The details of how this has been implemented in WP6, including how dynamic changes in the consent granted by the data subject are incorporated via the Data Gatekeeper, are described in Section 6. The Data Gatekeeper is also able to support dynamic changes to allow processing to protect vital interests or to pursue legitimate interests of the data controller. For example, the Data Gatekeeper can allow a suitably qualified (and authenticated) expert to assert vital interests, enabling processing to protect the data subject or another natural person. It can also support checks when a data subject registers to use services, verifying aspects such as the data subject’s age that may mean processing in pursuit of legitimate interests should be restricted.

The remaining cases from Table 5.1 do not involve rapid dynamic changes. A function that is needed to deliver the service will always be needed (as if it were not, it should not be present). A function that meets a legal obligation or a public interest can always be used, unless or until the law changes. That could happen during the lifetime of a system, but the process is slow enough that it can be covered by a redesign. It does not require changes to be managed at run-time. The use of Sticky Policies therefore addresses most dynamic changes associated with the legality of personal data processing under the GDPR.

The Data Protection Policy should include clauses that identify which legal basis is used for each of the functions of the service, related to its purpose. Where the legal basis is not consent, the Data Protection Policy should state any restrictions on processing, e.g. which laws require the controller to process data to meet a legal obligation, who can decide when processing is necessary to protect vital interests, or what measures are used to protect fundamental rights (including age restrictions) when processing in pursuit of legitimate interests.

5.5 Other Regulatory Constraints

We have already seen that when the data subject is a child, additional restrictions are imposed on the use of their personal data. This is covered explicitly under Article 8, which states that parental approval is normally needed before processing by consent of the data subject. Recital 38 also says that particular care must be taken when using personal data for marketing or profiling, or when a service is specifically designed for to be provided to children. RestAssured does not address these cases directly, although they could be addressed by extensions of RestAssured technologies.

Article 9 identifies special categories of data which are especially sensitive and/or afford greater potential for abuse (deliberate or otherwise). These categories cover emphdata revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership, and the processing of genetic data, biometric data for the purpose of uniquely identifying a natural person, data concerning health or data concerning a natural person’s sex life or sexual orientation.

Where data belongs to one of these categories, additional restrictions apply, for example:

• Processing by consent is permitted only if informed consent is granted, i.e. the data subject having been informed in a way comprehensible, still grant consent for the processing.

• Processing to protect vital interests is permitted without consent only if the data subject is physically or legally incapable of giving their consent.
• Processing for legitimate interests is permitted only for specific business purposes in which the use of special category data is necessary, e.g. in the field of employment, social security and protection, or by a not-for-profit body concerned with political, religious or trade union activities, etc.

• Processing in the public interest is limited to cases where this is proportionate to the aim pursued, and subject to suitable safeguards required by Union or Member State laws.

The handling of special categories of data is therefore somewhat affected by national law, and not only by the harmonised measures specified in the GDPR. Article 9(4) allows Member States to impose further conditions, including limitations, for processing of genetic data, biometric data, or data pertaining to health. This exception to the harmonization of regulations underpinning the single digital market recognises both the extremely sensitive nature of these types of data, and the prior existence of additional protection under national laws that could not reasonably have been withdrawn by the GDPR.

Although laws do not change fast enough to require a dynamic change management process, in a cloud application data sources and data processing and storage is not fixed:

• data sources (including the data subject) can move between countries;
• data analytics services can be halted and moved to data centres in different countries;
• stored data can be moved between data centres in different countries.

By virtue of Article 9(4), these movements between different jurisdictions can result in rapid changes in regulations that apply to data and processing, and thus in the compliance requirements for cloud-based applications.

It is beyond the scope of RestAssured to examine the restrictions imposed by different national regulations for these types of data. However, RestAssured does aim to support compliance with the GDPR, so the need to address different national regulations as one crosses a jurisdictional boundary is in scope. This has been addressed by the following measures:

• The knowledge base used to create SSM models in WP7 includes data classes representing types of data subject to national restrictions under Article 9(4), and a threat representing the need to address multiple regulations if it is inferred that such data crosses a jurisdictional boundary.

• The run-time model from WP5 includes the location (i.e. the jurisdiction) for each compute resource, and run-time monitoring supports detection of changes in location, and hence whether or not data sources and destinations are now in different jurisdictions.

These models are needed to address compliance in the PAYD scenario from WP8. In this scenario, a driver buys insurance from their insurer at a price that depends on their driving behaviour. For safer drivers, this leads to a lower insurance premium. To benefit from this service, the driver must allow collection and processing of several types of data:

• their name, address, age, any relevant medical conditions, history of driving offences, etc., i.e. data that would be required for any insurance scheme;
• data collected from their vehicle tracking where it is driven, and how it is driven, e.g. the speed, use of accelerator and brake functions, etc.
• data collected from the driver tracking their status, e.g. pulse rate, blink rate, etc., which indicates use of the vehicle when tired or stressed.
Some of this data is classified as biometric data, and so is subject to national regulations under the GDPR Article 9(4). Some Member States impose rigorous protections for this data, and in order that those protections should not be undermined by transferring it to another country, some impose the restriction that biometric data should not be exported even to other Member States. The safest way to meet national regulations is therefore to ensure that the biometric data does not cross a border.

The flow of data from the car to a cloud service is therefore only fully enabled when the car is in the same jurisdiction as the insurer’s data collection and analytics service. The process for handling any changes can be summarised as follows:

1) The run-time monitor detects when the car crosses a border to another Member State.
2) The run-time model is updated with its new jurisdiction.
3) The adaptation system requests a new risk assessment based on this change.
4) The run-time risk evaluator reports that there is a compliance threat from cross-border transfers of biometric data.
5) The adaptation system proposes a change, e.g. moving the data collection and analysis service for this driver to a data centre in the country they are now in.
6) The run-time risk evaluator reports whether this is acceptable, given that security may be less stringent.
7) The adaptation system implements the change by moving the service and telling the client application deployed in the car where its data should now be sent.
8) The run-time model is updated with the new jurisdiction of the service.

The sticky policy enforcement system detects the jurisdictional context for both the application client and the data collection service via the run-time model, as shown in Figure [11]. The change in jurisdiction for the car becomes evident in step (2), so the Data Gatekeeper should refuse access to allow the client to connect to the database when it next attempts to report more data. The change for the data collection service becomes evident at step (8), at which point the Data Gatekeeper can once again allow access.
6 Communicating Data Protection Policies across Enclaves

This section is about the data processing phase of the data lifecycle security. It details the case where data is executed on an enclave illustrating an integration with WP4.

6.1 Secure Enclave

A Trusted Execution Environment (TEE) is defined as a restricted security environment running in parallel with a Rich Execution Environment (REE) main operating system that may be Windows, Android, Linux ... Its purpose is to provide security services such as cryptographic key management, sensitive data processing, and rights management (Digital Rights Management). There are different hardware technologies to set up a TEE: in particular Intel SGX and AMD Enclave, that implement secure hardware enclaves.

A secure hardware enclave allows the isolation of certain elements of the software stack, creating private memory regions that are isolated from other applications and processes regardless of their privilege level including other user applications, the operating system, the hypervisor, etc. An enclave is therefore able to guarantee the isolation of a piece of software, even when the operating system or the hypervisor is malicious or corrupted.

The isolation provided by the enclave maintains the confidentiality of the data and code of the application and ensures their integrity.

6.2 Secure Data Processing Enforcement

In the Rest-Assured project, personal and confidential data are processed in Secure Enclaves, set up by AMD Enclave or Intel SGX. The interest is thus to obtain secure data processing without possibilities for an external process, even if launched with a higher level of privilege, to get access to the processed data.

As detailed in section 3, the data subject specifies his consent (or not) for data processing in the sticky policy. This consent can be under a context condition. This context condition can be related to the execution environment of the data and specifies an obligation to process the data on an enclave (see description of sticky policy structure in D6.1, Section 3.2).

Figure 6.1 illustrates an example of sticky policy where the data subject requires a processing of his data on an enclave.

In order to enforce such policies, Data Protection Enforcement Points are deployed upstream of an enclave(see Figure 6.2).

When a request for a personal data processing is issued by a service, it is intercepted by the Data Protection Enforcement Point. The request is forwarded to the Data Protection Decision Point that filters the access control sticky policies relevant to the query and computes the access rights to the stored data accordingly. The access control response is issued to the Data Protection Enforcement Point that enforces the decision and applies it to Secure Enclaves processing by blocking the request or sending it by listing the references of the authorized data. The enclave can then get access to the referenced data in the encrypted data base for processing.

6.3 Communicating Policy Changes Across Enclaves

Sticky policies can be separated into 2 logical parts, a content part that defines the policy and a reference part, which points to the data related to the policy. The content part is modifiable by the Data Subject associated with the policy and gives different access rights to the data pointed by the reference depending on its values.
Figure 6.1: Sticky Policy Partial Serialization

Figure 6.2: Secure Data Processing Enforcement
It should be noted that the confidentiality of sticky policies is not desired. In certain use cases, the Service Provider wishes to know the access rights of registered users in order to adapt its services and pricing (see PAYD use case description).

However, the integrity of sticky policies is of major importance. The Data Gatekeeper provides services to measure the integrity of security sticky policies, via policy signature algorithms that were described in D6.2, Section 4.2. The policy modification and deletion process and how it is managed by the Data Gatekeeper is also detailed in Section 4.3 of D6.2.

The impact of sticky policy changes is immediate for each request that occurs after the change. Indeed, the Data Protection Enforcement Point placed upstream of the enclave and the encrypted data base intercepts all the requests and forwards them to the Data Protection Decision Point that recalculates data access permissions and issues an updated decision based on the new access control policies stored in the sticky policy manager.

Thus, if between two queries an access control policy changes, its new content will be retrieved by the Data Protection Decision Point during the filtering step of relevant policies. After checking the signature, the updated content of the policy will be taken into account when calculating the access rights to the personal data as shown in Figure 6.3.

![Figure 6.3: Enforcing Changes Across Enclaves](image-url)
7 Data Gatekeeper Implementation

The Data Gatekeeper includes six components (see Figure 7.1): a Data Protection Contract Manager, a Sticky Policy Manager, a Data Protection Decision Point, a Data Protection Enforcement Point, a Context Manager, and an Authentication Manager.

Data Gatekeeper components are implemented in Java. The Data Protection Contract Manager, Context Manager, Sticky Policy Manager and Data Protection Decision Point are wrapped in a web service, accessible through a REST API. This API is used for interactions between Data Subjects and the Sticky Policy Manager to define access control policies. The API also allows a Service Provider to define its Data Protection Contract. Finally, this API is used for exchanges between the Data Protection Enforcement Point and the Data Protection Decision Point in order to exchange information on personal data access request and access decisions for fine grain access.

![Figure 7.1: General Architecture](https://restassuredh2020.eu//)

### 7.1 Context Manager

The Context Manager is a component based on semantic linked data technologies. The Context is implemented in RDF format, following an ontology structure. The context information is stored in a triplestore: Apache Jena TDB.

A context ontology is defined to describe the capabilities, specificities, and properties of the Rest-Assured system (for example, it is possible to describe the execution environment and in particular whether Secure Enclaves are provided. The context is separated into several parts:
• information about the runtime environment
• information on Data Consumer
• information about the Service

![Figure 7.2: Context Manager Operations](image)

Information on the Data Consumer comes from the authentication component that provides Data Consumer attributes and roles. Information about the runtime environment and information about the Service is provided by Adaptation and Risk Management. The context associated for the request is defined for each user. The context is thus defined and represented in a Named Graph. It is updated by SPARQL requests according to reconfigurations and adaptations of the Rest-Assured system.

When a request reaches the Data Protection Decision Point (see Figure 7.2), the context manager updates the context using SPARQL queries to the affected context Named Graphs. Based on [7] work, SPARQL ASK queries are extracted from the sticky policies and are applied to the associated Named Graphs.

A SPARQL ASK query is used to check the inclusion of an RDF graph in a Named Graph. These SPARQL ASK queries thus make it possible to define minimal contexts without which processing is not allowed. These contexts can notably be used to encode legal restrictions from the GDPR or geolocation constraints, secure processing constraints, etc.

Depending on the result of these SPARQL ASK queries, different filters are applied to the authorized data. Thus, some sensitive data processing may be allowed provided that the processes run in secure enclaves. The same data processing being limited to non-sensitive data in the case where secure enclaves are no longer available.
7.1.1 Context Manager API

The following endpoints are parts of the Context Manager. They are responsible for creating, updating and deleting Context Information, related to queries on personal data.

7.1.1.1 Creating Context

- POST
  - https://DATAGATEKEEPER-ADDRESS/decisionpoint/services/request/createContext

This endpoint generates a Named Graph related to a Data Subject and a Service ID in which context information is stored in RDF format.

7.1.1.2 Update Context

- POST
  - https://DATAGATEKEEPER-ADDRESS/decisionpoint/services/request/updateContext

This endpoint updates a Named Graph storing context information with up-to-date context information.

7.1.1.3 Delete Context

- DELETE
  - https://DATAGATEKEEPER-ADDRESS/decisionpoint/services/request/deleteContext

This endpoint deletes a Named Graph storing context information related to a Data Subject and a Service ID.

7.2 Authentication Manager

The Authentication Manager is responsible for authenticating Service Providers, that use and process personal data within the Rest-Assured system. The Authentication Manager also authenticates the Data Subjects which want to use registered Services specifying their data protection requirements.

In order to ensure the homogeneity and security of the Rest-Assured system, the authentication is federated between the RestAssured System and the service Providers. The DataGatekeeper, through its Authentication Manager, provides an identity federation system based on OpenID Connect.

OpenID Connect is an authentication standard build on top of the authorization protocol OAuth2.0. OpenID Connect implements federated authentication. OpenID Connect authentication uses OAuth2.0 to get a delegated authorization from the Data Subject to access his/her data. Then, the OpenID Connect server provides a back-url to go back to the Service after the authentication step.

In more detail, the authentication process of OpenID Connect defines 3 participants:

- The OpenID Provider is a server that can provide authentication of End-Users to a Relying Party
- The Relying Party is a client application delegating its End-Users authentication mechanism to an OpenID Provider
- The End-User is a participant willing to authenticate on the Relying Party
In the Rest-Assured system, End-Users are either Data Subjects or users of the Services that are registered on the Rest-Assured system. Relying Parties are Services registered and deployed on the Rest-Assured system. OpenID provider is implemented and configured using a Keycloak server (https://www.keycloak.org). The OpenID Connect authentication process is detailed in the sequence diagram of Figure 7.3. Its deployment in the RestAssured system is detailed in Figure 7.4.

![Figure 7.3: OpenID Connect protocol flow](image)

- The Relying Party receives a query without any token (or with invalid token) from an End User. The Relying Party redirects the End User to the OpenID Provider.
- The OpenID Provider authenticates the End-User. The OpenID Connect protocol does not depend on the authentication mechanisms in place. The choice of this mechanism is left to the OpenID Server. That can be FIDO, biometry, etc. In the Rest-Assured system, we rely on a login-password authentication.
- The OpenID Provider delivers a code to the End-User and sends the End-User back to the Relying Party.
- The End-User presents its code to the Relying Party.
- The Relying Party forwards the code to the OpenID Provider. The OpenID Provider verifies the code validity and delivers a token to the Relying Party. The token is composed of an id token, an access token and optionally a refresh token.
- The Relying Party forwards the token to the End-User.

Then the End-User embeds its token with every request. The Relying Party is able to authenticate him as long as the token is valid.
7.3 Data Protection Contract Manager

The Data Protection Contract Manager is responsible for the creation, signature, and storage of the data protection contracts in an XML format. In addition to the planned usages on personal data, data protection contracts include information about the Database structure and deployment within the Rest-Assured system. The structure of a data protection contract is represented in Figure 7.5.

7.3.1 Data Protection Contract Manager API

The following endpoints are parts of the Data Protection Contract Manager. They are responsible for registering Service Providers to the Rest Assured Data Gatekeeper data lifecycle manager. They should be accessible to the Service Providers in order for them to register the capabilities and usages of personal data. A Graphical User Interface is furnished to the Service Providers in order for them to register.

7.3.1.1 Display Data Protection Contract

- GET

- https://DATAGATEKEEPER-ADDRESS/dataprotectioncontract/services/serviceID/displayContract

This endpoint displays the content of a Data Protection Contract.
7.3.1.2 Build Template Contract

- POST
  - https://DATAGATEKEEPER-ADDRESS/dataprotectioncontract/buildContract

This endpoint get as input Post Parameters and builds a template for data protection contract in the GUI. This template will then be filled by the Service Provider in the generate contract endpoint.

7.3.1.3 Generate Contract

- POST
  - https://DATAGATEKEEPER-ADDRESS/dataprotectioncontract/generateContract

This endpoint can be based on the template from Build Template Contract endpoint. This endpoint generates a Data Protection Contract from Post Parameters. It checks the generated contract against a XSD scheme, and if valid, an XML signature is generated for the contract. The contract is then stored by the Data Protection Contract Manager.

7.4 Sticky Policy Manager

The Sticky Policy Manager is responsible for the creation, signature, and storage of the sticky policies.

The Sticky Policies are stored using semantic web technology. The Sticky Policies are thus stored as a graph of data, following the structure of an ontology.

The Sticky Policy Manager is a semantic component, composed by a triplestore and an interface allowing interacting with this triplestore. A triplestore is a high performance RDF store engine. Since we implement Sticky Policies as RDF format data, Sticky Policies are stored in the triplestore. We use Apache Jena TDB triplestore for the Rest-Assured project.
Here is a list of the main features of this component:

- Storage and retrieval of access control policies
- Persistence of the Sticky Policies in a triplestore: Apache Jena TDB
- Implementation of the Sticky Policy API allowing interacting with the triplestore, offering possibilities to create, update, delete, read, sign and filter security policies
- a GUI for the Data Subjects to modify their security policy

The Sticky Policy is generated for a Data Subject when he/she first registers to a specific service in the Rest-Assured system. A Sticky Policy derives its structure from the Data Protection Contracts between the Services that the Data Subject wants to give consent and the Data Gatekeeper. A sticky policy can be separated in 3 main components:

- a reference part, that links the policy to the related pieces of data within the Rest-Assured system
- a content part, that specifies the Data Subjects requirement in terms of use and processing of personal data
- a signature part, that ensures the integrity of the Sticky Policy

Semantic technologies are used to logically link a Sticky Policy to a piece of data belonging to a Data Subject. Sticky Policies are then stored as RDF Graph in a Triplestore. Semantic technologies are also used for computing access control decisions. A representation of the personal data is implemented as a RDF Graph, storing only a structure of the personal data, without information about the value of the data. Based on this representation, SPARQL queries are applied on Sticky Policies to extract relevant fields of the policy, related to a query on personal data. These fields are then processed by the Data Protection Decision Point, in order to compute a decision.

The Sticky Policy Manager also implements a signature mechanism in order to ensure integrity of Sticky Policies in the Apache Jena TDB Triplestore. A canonical serialization algorithm have been implemented as well as the procedure for signing a RDF Graph defined by the W3C Community Group (https://w3c-dvcg.github.io/ld-signatures/).

### 7.4.1 Sticky Policy Manager API

These endpoints are part of the Sticky Policy Manager component. They should be accessible for the Data Subject itself, as they allow to generate, update, delete Sticky Policies linked with personal data owned by the Data Subject. For convenience, some parts of the API can be accessed through a Graphical User Interface.

#### 7.4.1.1 Graphical Representation

- GET
  - https://DATAGATEKEEPER-ADDRESS/decisionpoint/dataSubject/dataSubjectID/displayPolicy

This endpoint displays a graphical representation of the security policy linked with the personal data of dataSubjectID.
7.4.1.2 Service Registry

- GET
  - https://DATAGATEKEEPER-ADDRESS/decisionpoint/listServices

This endpoint lists the services registered on the Data Gatekeeper.

7.4.1.3 Data Subject Creation

- POST
  - https://DATAGATEKEEPER-ADDRESS/decisionpoint/services/request/registerFirst

This endpoint creates a data subject as a graph with personal identifiers in the TripleStore of the Sticky Policy Manager.

7.4.1.4 Creating Sticky Policy

- POST
  - https://DATAGATEKEEPER-ADDRESS/decisionpoint/createFirstPolicy

This endpoint populates a security Sticky Policy with the privacy requirements of the Data Subject.

7.4.1.5 Updating Sticky Policy

- POST
  - https://DATAGATEKEEPER-ADDRESS/decisionpoint/updatePolicy

This endpoint updates a Sticky Policy.

7.4.1.6 Delete Sticky Policy

- DELETE
  - https://DATAGATEKEEPER-ADDRESS/decisionpoint/deletePolicy

This endpoint deletes a part or all of the Sticky Policy related to a Data Subject. The identity of the Data Subject must be passed as query parameter.

7.4.1.7 Signature Sticky Policy

- GET
  - https://DATAGATEKEEPER-ADDRESS/signPolicy

This endpoint sign the Sticky Policy related to a Data Subject. The identity of the Data Subject must be passed as query parameter.
7.4.1.8 Validate Sticky Policy

- **GET**

  
  https://DATAGATEKEEPER-ADDRESS/validatePolicy

This endpoint validates the signature of the Sticky Policy related to a Data Subject. The identity of the Data Subject must be passed as query parameter.

7.5 Data Protection Decision Point

The Data Protection Decision Point is a central component of the DataGatekeeper. It is responsible for making decisions on access to personal data in the Rest-Assured system. It is a component implemented in Java. The Data Protection Decision Point provides decisions on whether or not to access personal data. These decisions are evaluated for each access request forwarded by the Data Protection Enforcement Point.

The Data Protection Decision Point interacts with the Context Manager to update the context associated with the query, and deduces the corresponding filters and access conditions. For this, information is extracted from the authentication information of the Data Consumer provided by the Authentication Manager. Information is also extracted from the adaptation and monitoring data of the Rest-Assured system. A SPARQL Update query is built and then applied by the Context Manager on the Named Graphs of Context. It updates the context with the latest information on the query.

The Data Protection Decision Point also interacts with the Sticky Policy Manager. It builds a SPARQL query to extract the Named Graphs from the Sticky Policies concerned by the request. Then, the Sticky Policy Manager evaluates the access control policies by browsing the graphs associated with the Sticky Policies in order to extract the access rights satisfying the up-to-date context conditions and user preferences. The Sticky Policy Manager then retrieves the pointers to the authorized data, and forwards them to the Data Protection Decision Point.

The final decision is fine-grained; for a given request, it is possible to allow access to all possible subsets of data in a Data Subject without granting permission to all of its data. This decision is forwarded from the Data Protection Decision Point to the Data Protection Enforcement Point. The latter being responsible for the application of the decision.

7.5.1 Data Protection Decision Point API

The following endpoints are part of the Data Protection Decision Point component. They are the core intelligence of the Data Gatekeeper, responsible for taking decision on processing and treatment of personal data, depending on context information and personal/individual requirements on privacy. Thus, allowing a fine grained access control respecting privacy requirements of every individual for which personal data are processed.

7.5.1.1 Aggregated Authorization with Context

- **GET**

  

This endpoint delivers the set of authorized entries for a query through a service. This endpoint is called by the Enforcement Point, implemented as a Query Gateway component, specifying the requesting service and declared data usage as serviceID and usageID path parameters, as well as the set of concerned entries in the
SQL Tables as query parameters. This endpoint also asks the Context Manager to update and compute the related context information. The endpoint answers with the list of authorized entries (may be empty) associated with the registered primary identifier of the service serviceID databases, depending on the computed context. The list of concerned data must be sent as query parameters.

### 7.5.1.2 Specific Authorization with Context

- **GET**

  - https://DATAGATEKEEPER-ADDRESS/decisionpoint/services/serviceID/usage/usageID/dataSubject/dataSubjectID/askAuthorizationContext

This endpoint delivers a signed JWT for a query on a specific data subject. This endpoint is called by the Enforcement Point, implemented as a Query Gateway component, specifying the serviceID, dataSubjectID and usageID as path parameters. This endpoint also asks the Context Manager to update and compute the related context information. The endpoint answers with a signed JWT stating which data is allowed for the data subject under the computed context. Thus implementing fine grained authorization. In the case where there is no authorized data, an empty json object is sent.

### 7.5.1.3 Simple Authorization

- **GET**

  - https://DATAGATEKEEPER-ADDRESS/decisionpoint/services/serviceID/usage/usageID/dataSubject/dataSubjectID/simpleDecision

This endpoint delivers a Grant or Deny response for a query. This implement a coarse grained access control.

### 7.6 Data Protection Enforcement Point

The Data Protection Enforcement Point resides in front of the Data Stores and acts as a reverse proxy for incoming queries. It forwards information about the query to the Data Protection Decision Point. As the Data Protection Decision Point is wrapped in a web service, reachable via a REST API over HTTP, the Data Protection Enforcement Point sends HTTP queries to the Data Protection Decision Point and interprets and enforces the decision provided in the response. The Data Protection Enforcement Point is closely related to the Data Store it protects. In the Rest-Assured system, SQL queries are used. Thus, the Data Protection Enforcement Point implements SQL query rewriting in order to enforce the decision sent by the Data Protection Decision Point.
8 Application in RestAssured Use Cases

8.1 Data Lifecycle Security in the PAYD Use Case

8.1.1 Overview and Background

During the second half of the project, the PAYD Use Case expanded its original scope to encapsulate an additional challenge - cross-border service continuity including a mix of flowable (geolocational) and non-flowable (biometric) data. In the case of the GDPR, the free flow of personal data is permitted between member states, and may be further extended for transfer to third countries or international organisations pursuant to the requirements put forth in Articles 44-50. The addition of biometric data in the use case highlights a further case - the intersection of the GDPR with national level legislation. While GDPR Art. 9(1) prohibits the processing of special category data (including biometric data), this can be exempted in the case where the data subject provides their consent for such processing - except for cases where the data subject is not empowered to do so by applicable national legislation.

Within the PAYD Use Case, cross-border data transfers and accesses are dealt with in the following way:

1. Determining whether biometric data is permitted to leave the country of origin.
2. Determining whether the destination country provides data protection adequacy for such data.
3. Determining whether the data subject has granted their consent for such a transfer to occur.

A key challenge for Data Lifecycle Security is in being able to first capture this level of access control, while also providing decision and enforcement capabilities at the appropriate level of granularity (e.g. in the Cloud or at the Edge) in order to facilitate compliance in the given environmental and operational context.

Note that a comprehensive description of this use case is out of scope for this document. A more detailed explanation can be found in D8.3[13], with a detailed analysis of results being provided in the forthcoming D8.4.

An overview of the overall system architecture is provided in Figure 8.1.

8.1.2 Role of the Data GateKeeper

The Data GateKeeper plays a key role in determining what types of data accesses are allowed or rejected across the PAYD components.

Situational awareness and contextualization of the use case environment depends on account for both internal (through adaptation to the backend components) and external factors (through changes in e.g. end-user consent, or changes in physical location of the data subject).

8.1.3 Integration with Other RestAssured Components

Within the PAYD use case, the role of the adaptation component and the Data GateKeeper are intrinsically linked - the Data GateKeeper for the codification of, and enforcement of policy, in a variety of contexts, and the adaptation component for making targeted adaptation recommendations to the PAYD API Gateway depending on the results of risk assessment. The proposed adaptation changes may require changes in the overall deployment itself, functionality gating within the backend components, or in partial reconfiguration of the user-facing front-end application. Each of the proposed adaptations may further require adaptation of the underlying policy being enforced by the Data GateKeeper in order to ensure compliance with the implemented adaptation scheme.
8.2 Data Lifecycle Security in the HPC Use Case

8.2.1 Overview and Background

Within the HPC use case, end-users submit data for processing into an HPC Cluster with an exposed Cloud frontend for supporting data ingress and workload management, secured by RestAssured technologies. In this scenario, documents of unknown sensitivity are bulk transferred for analysis and classification by a document classification algorithm, prior to subsequent processing in classification-appropriate data pipelines. Each of the classification levels, in turn, place different requirements on the types of processing that can be carried out, and the way in which data must be handled - whether this can be encrypted both at rest and in-processing, logging of all accesses and stages of processing for subsequent auditing and forensic log analysis, etc.

This scenario aims to highlight a typical deployment scenario in Cloud - one where:

- specific resources required by a given workload are finite and may change in availability
- the cost of resource utilization varies depending on resource scarcity and availability
- the threat level to the workload changes depending on whether the workload is deployed into a single or multi-tenant environment

Note that a comprehensive description of this use case is out of scope for this document. A more detailed explanation can be found in D8.3[13], with a detailed analysis of results being provided in the forthcoming D8.4.

An overview of the overall system architecture is provided in Figure 8.2 below:
8.2.2 Role of the Data GateKeeper

The Data GateKeeper is responsible for determining the appropriate level of access and required level of protection for a given classification level, and can further outline the environmental criteria that must be matched in order for a given level to be processed.

8.2.3 Integration with Other RestAssured Components

The role of Adaptation in this case is two-fold:

1. monitoring for changes in document sensitivity as a result of dynamic data classification; and
2. monitoring for changes in tenancy on the deployed infrastructure.

As many of the protection mechanisms available come at a cost, the workload relies on the run-time monitor and adaptation mechanism to ensure that appropriate protections are put in place at such a time that the threat level raises. Further, when the threat level decreases, the requirements for the workload can be relaxed and shifted back to resources subject to less overall contention - freeing up the highly contended resources for those with a more immediate pressing need.
9 Conclusion

This document elaborates the design time phases of the data lifecycle security management that were detailed in D6.1 and develops the runtime phases that are not covered by D6.1.

The design time phases are implemented and extended with a support of data protection legislation compliance based on data protection policies that are used as input for data protection contracts. The document details an approach for assisted generation of data protection policies using textual patterns. The proposed approach simplifies the problem of formulating policies for complex systems, supports service providers to provide a transparent and comprehensible data usage policy to data controllers leading to simpler, more comprehensible statements for data subjects.

The development of the runtime phases is reflected by the implementation of the context-based approach, the compliance checking and the change management during the whole lifecycle that is supported by an integration with the adaptation platform of WP5 and risk analysis of WP7. The data protection tools from WP6 address dynamic changes in consent originating from the data subject, and support other dynamic changes such as the need to protect vital interests that may mean data should be more accessible, or to introduce additional safeguards for minors that may mean data should be less accessible.

The implementation of a federated identity and authentication system in the Data Gatekeeper takes into account the distributed cloud environment and multi-party dimension by providing trustworthiness for data subjects and flexibility and autonomy to the various service providers.

The Data Gatekeeper now supports the communication of data protection policies across enclaves and all the main access control requirements for continuous compliance with GDPR Article 6, regardless of which legal basis allows personal data processing.

In the future, the Data Gatekeeper and the data lifecycle security methodology will be extended to be deployed in a fog computing and 5G environment.
Bibliography


