The research leading to these results has received funding from the European Community’s H2020 research and innovation programme under grant agreement n° 731678.
RestAssured Consortium

Project Number: 731678
Project Title: RestAssured - Secure Data Processing in the Cloud
Deliverable Type: External Report

Title of Deliverable: First RestAssured Handbook
Nature of Deliverable: External Report
Dissemination Level: Public
Contractual Delivery Date: 31 December 2018
Actual Delivery Date: 31 December 2018
Contributing WPs: All
Editor(s): John Boyle
### Contributors

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Boyle</td>
<td>OCC</td>
<td>Sections 0.1, 1, 3, 4</td>
</tr>
<tr>
<td>James Greig</td>
<td>OCC</td>
<td>Section 2.3.2</td>
</tr>
<tr>
<td>Zoltan Mann</td>
<td>UDE</td>
<td>Section 2.3.1</td>
</tr>
<tr>
<td>Nazila Gol Mohammadi</td>
<td>UDE</td>
<td>Section 1.2</td>
</tr>
<tr>
<td>Paul Mundt</td>
<td>Adaptant</td>
<td>Sections 2.3.3, 3</td>
</tr>
<tr>
<td>Eliot Salant</td>
<td>IBM</td>
<td>Sections 1.1, 2, 3</td>
</tr>
<tr>
<td>Cyrille Piatte</td>
<td>Thales</td>
<td>Sections 1.3, 2, 3</td>
</tr>
<tr>
<td>Toby Wilkinson</td>
<td>ITInnov</td>
<td>Sections 1.2, 2, 3</td>
</tr>
</tbody>
</table>
## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>October 2018</td>
<td>Initial version</td>
</tr>
</tbody>
</table>

Contents

List of Figures ........................................................................ 7
List of Tables ......................................................................... 9
  0.1 Introduction .................................................................. 10

1 Overview of RestAssured ...................................................... 11
  1.1 Overview - Update ...................................................... 12
  1.2 Risk Models ............................................................... 13
    1.2.1 The CSAP Tool ...................................................... 14
    1.2.2 System Security Modeller ........................................ 18
  1.3 Data Lifecycle ................................................................ 22

2 How to Build a Business Offering based on RestAssured Technology ........................................ 27
  2.1 Introduction .................................................................. 27
  2.2 Components .................................................................. 28
    2.2.1 TruCE framework .................................................. 28
    2.2.2 Flask .................................................................... 28
    2.2.3 System Security Modeller ........................................ 28
    2.2.4 Risk@Runtime ....................................................... 28
    2.2.5 Thales Components ................................................ 29
  2.3 Building Solution Examples ............................................. 31
    2.3.1 Enabling run-time data protection assurance .................. 31
    2.3.2 Integrating the SCANT Use Case ......................... 32
    2.3.3 Integrating the Pay-As-You-Drive Use Case ............... 37

3 RestAssured Handbook V2 .................................................... 41

4 Conclusions ....................................................................... 42

A Appendices ....................................................................... 43
  A.1 References to Open Source Software ............................... 44
    A.1.1 Parquet encryption ............................................... 44
    A.1.2 Flask .................................................................... 44
    A.1.3 Keycloak .............................................................. 44
    A.1.4 Apache Jena ......................................................... 44
    A.1.5 Jersey ................................................................. 44
A.2 References to API Documentation

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.2.1 Data Gatekeeper: Data Protection Decision Point</td>
<td>45</td>
</tr>
<tr>
<td>A.2.2 Data Gatekeeper: Sticky Policy Manager</td>
<td>45</td>
</tr>
<tr>
<td>A.2.3 Data Gatekeeper: Data Protection Contract Manager</td>
<td>47</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1.1: Risk assessment process ................................................................. 13
Figure 1.2: Editor mode of the CSAP-tool ...................................................... 14
Figure 1.3: Adding a Cloud Element and defining its name ......................... 15
Figure 1.4: Equality of the name and instance type of an added Cloud Element ........................................................................................................... 16
Figure 1.5: Instantiation of a Cloud Element of the instance type Data in a ReAs-CSAP ..................................................................................................... 16
Figure 1.6: Representation of the instantiated Cloud Element of the instance type Data ........................................................................................................ 17
Figure 1.7: Definition of Properties of the instantiated Cloud Element of the instance type Data .................................................................................. 17
Figure 1.8: Asset palette, canvas, and relationships ....................................... 18
Figure 1.9: Misbehaviour impact levels, likelihoods, and risk levels .......... 19
Figure 1.10: Trustworthiness levels ................................................................. 19
Figure 1.11: Remaining threats in the SCANT use case after applying standard controls .................................................................................................... 20
Figure 1.12: Finding the root cause ................................................................. 21
Figure 1.13: Treating the root cause ............................................................... 22
Figure 1.14: Treating the GDPR compliance threat .................................... 23
Figure 1.15: Data Protection Contract Initialisation ...................................... 23
Figure 1.16: Data Protection Contract Specification ...................................... 24
Figure 1.17: Data Protection Contract Signature .......................................... 24
Figure 1.18: Change Preferences .................................................................. 25
Figure 1.19: Display Authorization ............................................................... 25
Figure 2.1: SCANT Software Architecture .................................................. 33
Figure 2.2: SCANT Implementation Overview ............................................ 34
Figure 2.3: SCANT User Interface ............................................................... 36
Figure 2.4: PAYD Conceptual Data Model .................................................. 38
Figure 2.5: PAYD Software Architecture .................................................. 40
List of Tables

Table 2.1: Components of the PAYD use case testing environment ................................. 39
0.1 Introduction

This report presents V1 of the RestAssured handbook: an online, multi-media handbook that explains how to use the RestAssured solutions in open source as well as commercial cloud environments.

The purpose of each of the RestAssured components is described in outline together with links to the RestAssured code libraries; open-source components; licencing agreements.

The handbook includes a set of use case descriptions designed to show how to apply and validate the RestAssured technologies. The Handbook therefore explains in a hands-on fashion how the RestAssured solutions may be used and will thus be an important means to achieve uptake and impact in practice.

Specifically, the handbook covers the following: section 1 is an overview of RestAssured for those new to the project objectives, technologies and delivery platforms. This is accompanied by a set of project videos, screenshots or descriptions in section 0.1. These cover:

- an overview of the RestAssured concept illustrated through the Social Care use case,
- an introduction to the RestAssured risk models and risk modelling tools,
- an overview of the individual RestAssured components.

Section 2 explains how to build a RestAssured business solution and offering. Each of the technical components is first described and then the application of those components to the business cases is summarised. For each use case we describe the architecture of the solution (as applied to the use case) and how the solution is actually implemented including clear reference to the code libraries and how they are called.

The handbook includes a set of appendices which reference the open source libraries used and the RestAssured APIs.

Because this is V1 of the handbook, section 3 explains the main additions to the handbook which we expect to make in the final 12 months of the project.
1 Overview of RestAssured

RestAssured provides solutions to specific technical concerns of data protection in the cloud (such as geo-location restrictions on personal data), which are imposed by the dynamic, multi-stakeholder and decentralized nature of federated cloud systems. These concerns mean that privacy and security by design approaches are no longer sufficient, due to uncertainty at design time of how the cloud and privacy requirements may dynamically evolve and change at run time. To this end, RestAssured provides novel mechanisms and cloud architectures for the runtime detection, prediction and prevention of data protection violations.

RestAssured assures the protection of sensitive business and citizen data in the cloud by combining four pillars of innovation: (1) combination of fully homomorphic encryption to process data without decryption with cloud enablement of SGX hardware for protected data processing, (2) sticky policies for decentralized data lifecycle management, (3) models@runtime for data protection assurance, and (4) automated risk management for run-time data protection.

RestAssured solutions are being demonstrated through three use cases driven by project partners; High Performance Computing for commercial enterprises; Pay As You Drive usage based insurance; and self-directed Social care for vulnerable adults and social care providers.

The RestAssured Handbook explains, in a hands-on fashion, how the RestAssured solutions may be used and is thus an important means to achieve uptake and impact in practice.
1.1 Overview - Update

RestAssured has released to YouTube a short video which explains the main concepts of RestAssured through the context of its social care use case.

This demonstration video, “RestAssured In Action: An example use case”, can be found at: [https://www.youtube.com/watch?v=z9Mu4rDwdy8](https://www.youtube.com/watch?v=z9Mu4rDwdy8)
1.2 Risk Models

For performing a risk assessment, it is necessary to obtain all relevant information about the context of the considered cloud computing service. The context analysis of a cloud computing service and its underlying system is a difficult task because of the variety of different types of information that have to be considered. This context information includes: i) Legal, regulatory and/or contractual requirements that are relevant for a cloud computing service (indirect stakeholders), ii) relations to other involved cloud computing services, iii) high-level cloud system components that support the involved cloud computing services, iv) data that is processed by the cloud computing services and v) stakeholders that interact directly with the cloud computing services and/or the underlying cloud system components.

RestAssured design-time risk assessment methodology first starts with a pattern-based approach for the context analysis of cloud computing services. Then, using a model-based approach RestAssured support the identification of threats and controls. Our context pattern contains elements that represent the above-mentioned types of context information. The elements of our pattern conform to the General Data Protection Regulation. Besides the context analysis, our pattern supports the identification of high-level assets. Additionally, our proposed pattern supports the documentation of the scope and boundaries of a cloud computing service conforming to the requirements of the ISO 27005 standard (information security risk management). The context analysis is the prerequisite for threat and control identification that are performed later in the risk management process.

The modelling of the context of a cloud computing service and its supporting cloud computing system represents the basis for a risk assessment (see Figure 1.1). Here, the context defines the scope and boundaries that are relevant for performing a risk assessment. Furthermore, the context provides necessary information that has to be considered during the risk assessment. On these assets the further steps of the risk assessment are performed.

<table>
<thead>
<tr>
<th>Context definition</th>
<th>Risk Assessment</th>
<th>Risk Treatment</th>
</tr>
</thead>
</table>

Figure 1.1: Risk assessment process

Definition of the context is needed for performing a risk assessment for a system. This context definition has to represent information about the external and the internal context of a cloud computing service. With respect to the external context, it is necessary to identify all the requirements that are indirectly relevant for the provided service and the corresponding system.

Work package 7 provides the risk assessment methodology of RestAssured. The two design-time risk assessment tools involved in risk assessment methodology are:

1. The Cloud System Analysis Pattern (CSAP) for establishing scope and context analysis of a system (supports Context Definition in Figure 1.1)
2. The System Security Modeller (SSM) for identifying threats and controls based on design-time system model (supports Risk Assessment and Treatment in Figure 1.1)
CSAP is a high-level context-oriented approach that employs patterns to identify the relationships between the system and the stakeholders (cf. D7.1). Here stakeholders include Data Subjects, Data Controllers, and Cloud Providers. CSAP represents a pattern for defining the context of a cloud computing service. Using the CSAP ensure that crucial information is not overlooked, e.g. by ensuring that all potentially relevant asset types are considered. For enabling a use of the CSAP in RestAssured the CSAP has been extended to include a new pattern. This new pattern of the CSAP is called RestAssured-Cloud System Analysis Pattern (ReAs-CSAP).

SSM employs graph-based models to model the assets of a system and the relationships between them (cf. D7.1). A domain specific catalogue of threat patterns captures the possible threats within a domain, and through pattern matching the specific threats in a system model can be identified. Moreover, by specifying the trustworthiness levels of certain attributes of system assets, the likelihood of those threats can be computed. This includes two mechanisms by which the effects of threats can be propagated: automatic secondary effect chaining, and loss of trustworthiness effects (cf. D7.1). Finally, by specifying the impact levels for primary asset misbehaviours, risk levels can be computed.

The two approaches have been implemented and described in detail in the CSAP (cf. D7.1, Section 4.1) and SSM (cf. D7.1, Section 4.2) tools, respectively.

1.2.1 The CSAP Tool

This tool provides a graphical editor that supports the creation and instantiation of Cloud System Analysis Patterns (CSAP). To this end, the tool provides two editors, namely the Designer-editor and the User-editor. These editors are explained in the following sections.

![Figure 1.2: Editor mode of the CSAP-tool](image)

1.2.1.1 The Designer-editor

The Designer-editor of the CSAP-tool enables designers to create specific Cloud System Analysis Patterns (CSAP). The designers can build a specific cloud system analysis pattern in two different ways:
In case of starting with an empty CSAP, designers can create their own CSAP by adding appropriate types of \textit{Indirect Stakeholders}, \textit{Direct Stakeholders} and \textit{Cloud Elements}. Furthermore, associations between Direct Stakeholders and Cloud Elements as well as between Direct Stakeholders among each other can be created. If the original CSAP is used as basis, already existing CSAP-elements can be modified and/or deleted.

Figure 1.3 shows the definition of a new Cloud Element that is added to an empty CSAP. In the designer-editor the name of a CSAP-element can be as the same as its instance type (see Figure 1.4).

1.2.1.2 The User-editor

In the \textit{User-editor}, any defined pattern can be instantiated. The instantiation of our ReAs-CSAP is described in D7.1. Figure 1.5 shows the instantiation of a Cloud Element of instance type \textit{Data} during the instantiation of the ReAs-CSAP. This Cloud Element instance specifies customers’ data and therefore is named \textit{Customer Data}.

The instantiation of the Cloud Element is marked by displaying the type name \textit{Data} surrounded by angle brackets under the name of the Cloud Element (see Fig. 1.6). For a subset of the properties of the instantiated Cloud Element, the corresponding values can be specified in the according dialogue during its instantiation (see Fig. 1.5). For all properties, except for the \textit{instance type}, the corresponding values can be assigned in a property panel (see Fig. 1.7).
Figure 1.4: Equality of the name and instance type of an added Cloud Element

Figure 1.5: Instantiation of a Cloud Element of the instance type Data in a ReAs-CSAP
Figure 1.6: Representation of the instantiated Cloud Element of the instance type Data in a ReASCAP

Figure 1.7: Definition of Properties of the instantiated Cloud Element of the instance type Data
1.2.2 System Security Modeller

System Security Modeller (SSM) is a web-based tool for performing design-time risk assessment of systems.

1.2.2.1 Constructing the Initial Model

The tool has a canvas on which the user can create a model of the system by dragging assets from a palette [Fig. 1.8 – (1)] on to the canvas [Fig. 1.8 – (2)] and adding the relationships between them [Fig. 1.8 – (3)]. Once this process is complete SSM applies machine reasoning techniques to:

- Check that the model is syntactically correct.
- Added certain inferred assets that exist by virtue of the relationships between the assets. For example this can include logical entities like paths between networks.
- Find threats that exist within the model using a knowledge base of threat patterns.

![Figure 1.8: Asset palette, canvas, and relationships](image)

1.2.2.2 Identifying Primary Assets

The next step is to identify the primary assets in the system, and specify the impact levels for the possible misbehaviours of those assets. By default SSM assigns relatively low impact levels to asset misbehaviours. This is entirely appropriate for supporting assets (as per ISO 27005), but not appropriate for the primary assets of the system. In Fig. 1.9 the Subject Data has been identified as a primary asset, and Loss of Confidentiality has been given a High impact level [Fig. 1.9 – (1)].
1.2.2.3 Applying System Specific Knowledge

The threats identified by SSM have likelihoods, and these in turn determine the likelihood of the asset misbehaviours that they cause [Fig. 1.9 – (2)]. The combination of the impact level of an asset misbehaviour, and the corresponding likelihood of that misbehaviour, determine the risk of that misbehaviour [Fig. 1.9 – (3)]. (See Section 3.2.5 of D7.1 for details of this calculation.)

The threat likelihoods are determined by the trustworthiness of assets, or more specifically, by the trustworthiness levels of certain attributes of those assets. Again, SSM chooses default values for these attributes, but the user can adjust these based upon their understanding of the system they are modelling. For example, in Fig. 1.10, which is taken from a model of the SCANT use case (Section 2.3.2), the AMI LAN is provisioned in a public cloud. As this may be shared with other tenants outside of the control of the SCANT operator, the user trustworthiness level is set to Low [Fig. 1.10 – (1)].
1.2.2.4 Adding Standard Security Measures

At this stage SSM will likely have identified a large number of potentially high risk threats. The next step is to systematically apply standard security measures:

- Software patching of all hosts (physical or virtual).
- Authenticate and encrypt all client-service communications.
- Use encrypted storage for the sensitive data.

In Fig. 1.11 we have applied these controls to the SCANT use case model. SSM’s reasoning determines that this addresses most of the threats, but leaves one high risk security threat [Fig. 1.11 – (1)], and one GDPR compliance threat [Fig. 1.11 – (2)].

Figure 1.11: Remaining threats in the SCANT use case after applying standard controls
1.2.2.5 Treating High Risk Threats

The high risk security threat can be examined in the Threat Explorer [Fig. 1.12 – (1)], where we see that this is a primary threat. Primary threats have entry points corresponding to trustworthiness attributes of the involved assets (see Section 3.2.2 of D7.1 for more details). Here the relevant trustworthiness attribute is the trustworthiness of the users of the Spark DB. Something is driving the trustworthiness level down from Very High to only Medium. We can investigate this further using the Misbehaviour Explorer [Fig. 1.12 – (2)], where we see another threat that is the root cause of the threat that we are investigating [Fig. 1.12 – (3)].

![Figure 1.12: Finding the root cause](image)

The root cause threat arises from the fact that the AMI Server is provisioned in a public cloud. The management of the server is outside the control of the SCANT operator, and thus the data processed by the Spark DB hosted on the AMI Server is potentially exposed. The available control strategy is run the SparkDB in a secure enclave on the AMI Server [Fig. 1.13 – (1)].
1.2.2.6 Treating GDPR Compliance Threats

The GDPR compliance threats can be examined in the Compliance Explorer [Fig. 1.14 – (1)], and each individual threat examined in turn [Fig. 1.14 – (2)]. For the SCANT use case, SSM has identified that the Spark DB may process the Subject Data in a way that is not compliant with the GDPR. The available control strategy is to apply Sticky Policies to the Subject Data, and enforce them at the Spark DB [Fig. 1.14 – (3)].
Figure 1.14: Treating the GDPR compliance threat
1.3 Data Lifecycle

In the RestAssured project, 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 such as 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. The objective of the Data Gatekeeper is to securely manage the complete data lifecycle in a decentralized cloud setting.

The Data Gatekeeper works as follows: Service provider offers a contract, specifying its service and the constraints it applies to the data. In parallel, the data subject specifies its requirements on the processing of its data. Both specifications are combined and encoded in a sticky policy. The sticky policy is attached to the data. The security policy is thus consulted before each use of the data.

The Data Gatekeeper is composed of 5 logical components:

1. The Data Protection Contract Manager is responsible for the registration of the Service providers, in Data Protection Contracts. It collects the type of data that is needed by the Service Providers and their usage (see figures 1.15 and 1.16). The Data Protection Contracts between the Data Gatekeeper and the Service Providers are generated, signed and stored, as shown in figure 1.17.

Figure 1.15: Data Protection Contract Initilisation
2. The Sticky Policy Manager component allows a Data Subject to register its preferences on the processing of its personal data, as shown in figure 1.18. In order to specify the possible options for data processing, the structure of the Data Protection Contract of the requested service is used. The Data Subject can verify the authorizations that he/she delivers to a service as seen in figure 1.19 and update these preferences. These preferences are translated into sticky policies that are logically bounded to
personal data and checked before processing the data. The Sticky Policy Manager is able to sign and store sticky policies.

3. The Data Protection Decision Point component is responsible for combining the various organizational access control policies and sticky policies in order to grant or deny the processing of the data.
The decision is forwarded to the Privacy Enforcement Point component.

4. The Data Protection Enforcement Point component is responsible for intercepting the request for personal data from Services, forwarding the request to the Data Protection Decision Point component and applying the decision made by this component. A Data Protection Enforcement Point is locally deployed for each Service, in front of each databases storing personal information.

5. The Authentication component is responsible for authenticating a data consumer requesting data through a service. When a data consumer logs in on a service with authentication delegation, he is directed to the Authentication component. He logs in on the Authentication component and is redirected to the service, being authenticated. The Authentication component is an authentication service, using an implementation of the OpenID Connect protocol.
2 How to Build a Business Offering based on RestAssured Technology

2.1 Introduction

RestAssured offers a blueprint to allow third parties to create a Cloud infrastructure which both offers a high degree of security for hosted applications, as well as giving Cloud infrastructure and Service Providers a means to help meet GDPR requirements. Additionally, a number of RestAssured components such as the Risk Analysis modeler (CSAP) can be used as standalone tools.

This handbook will serve as a guide to create a RestAssured-like system, and is meant to be used in conjunction with the more technical and detailed project documentation which can be found on the project website, https://restassuredh2020.eu/publications.

A high-level view of the architecture and the individual components is presented in the D3.2 and D3.3 High Level Architecture and Methodology deliverables. These documents should be considered as a starting point to gain a conceptual understanding of the goals of the RestAssured system and the interactions between the software components.

The actual programming interfaces (APIs) or low-level specifications for each component are described in the scientific reports for each technical work package, namely D4.1 for Secure Enclaves, D5.1 for runtime data protection assurance, D6.1 for data lifecycle security management, and D7.1 for the security and privacy monitoring component.

Some components have been implemented either in full or in part as open source and are publicly available. References to project-produced open source can be found in Appendix A.1. For those components that are proprietary, developers can either create alternative versions based on the published specifications, or contact RestAssured to learn about licensing possibilities.
2.2 Components

2.2.1 TruCE framework

Owner: IBM
License: Apache License 2.0
Link: [https://github.com/IBM/sgx-trust-management](https://github.com/IBM/sgx-trust-management)
Purpose:
Intel’s SGX enclaves add a layer of hardware-based security to cloud computing. However, working with SGX is difficult, and requires a high degree of expertise. To simplify this process, and enable the use of SGX hardware in RestAssured-like platforms, IBM developed Trust Management Framework (or TruCE for short - ‘Trust in Cloud Enclaves’). TRuCE handles all aspects of remote attestation and secret delivery process in Intel SGX enclaves. The framework enables application developers to focus on the application code, performing attestation by a simple API call. Additional calls are available for easy-to-use enclave sealing and secret passing to enclaves.

2.2.2 Flask

Owner: 3rd party open source
License: BSD [http://flask.pocoo.org/docs/1.0/license/#flask-license](http://flask.pocoo.org/docs/1.0/license/#flask-license)
Purpose:
Web server framework for SCANT UI. Easy to use and flexible

2.2.3 System Security Modeller

Owner: IT Innovation
License: Defined in Final version of Handbook
Link: Defined in Final version of Handbook
Purpose:
System Security Modeller (SSM) is a tool for performing a risk assessment of a system at design-time (i.e. before deployment) in line with ISO 27005. SSM is a web based application with a canvas onto which a user can drag icons representing the assets in a system, and these can be linked together to express how those assets are related. The result is a design-time model of the system. SSM can perform a detailed risk analysis of a design-time model, identify threats to assets, and the consequent risks to the system. The user is able to explore the model to identify the root causes of identified risks, and apply security controls to eliminate or mitigate the threats from which the risks arise. SSM recalculates the risk levels to take account of the security controls the user has selected, and the process is repeated until the risk level is acceptable to the user. The resultant set of security controls (and the assets to which they must be applied) constitutes a detailed set of security requirements that the deployed system should meet.

2.2.4 Risk@Runtime

Owner: IT Innovation
License: Defined in Final version of Handbook
Link: Defined in Final version of Handbook
Purpose:
Risk@Runtime is a software component whose primary objective is to perform risk analysis of systems at runtime (i.e. during the operation of the system). It is deployed as a software service, usable via a set of REST APIs. These allow initial design time risk models (created using the System Security Modeller SSM) to be uploaded (and updated dynamically) and set the initial context for the running system. Subsequent changes to the system (and/or environment) will trigger potential adaptations of the system’s behaviour e.g. to increase performance or maintain dependability – Risk@Runtime analyses every potential adaptation (input via the risk analysis API) in terms of a new runtime model of the system and calculates the new risk levels of this future state. The component can greenlight the suggested adaptation if it meets an acceptable risk level, or highlight where threats are greatest and need to be further mitigated by an adaptation.

2.2.5 Thales Components

Several assets of the Data Gatekeeper has been developed and used. They are wrapped in REST Web Services, using Jersey ([A.1.5]) framework. They must be deployed in trusted environments.

2.2.5.1 Data Protection Contract Manager

Owner: Thales  
License: To be defined  
Link: Not open source  
Purpose:  
The Data Protection Contract Manager is responsible for the registration of the Service providers, in Data Protection Contracts. It collects the type of data that is needed by the Service Providers and their planned usage. The Data Protection Contracts between the Data Gatekeeper and the Service Providers are generated, signed and stored.

2.2.5.2 Sticky Policy Manager

Owner: Thales  
License: To be defined  
Link: Not open source  
Purpose:  
The Sticky Policy Manager component is responsible for the registration of the Data Subject. It collects the data subject security preferences through a Graphical User Interface. The Sticky Policy Manager component translates the data subject requirements into Sticky Policies, and bounds them to the personal data. The Sticky Policies are generated in RDF (Resource Description Framework) Format, using the framework Apache Jena ([A.1.4]), released under Apache License 2.0. This component is also responsible for signing and storing the Sticky Policies, in a TripleStore. The TripleStore is also furnished by the framework Apache Jena.

2.2.5.3 Data Protection Decision Point

Owner: Thales  
License: To be defined  
Link: Not open source  
Open Source Alternatives: Authzforce: [https://github.com/authzforce](https://github.com/authzforce)  
Purpose:  
The Data Protection Decision Point component is responsible for extracting and combining the organizational access control policies and sticky policies. It will output a response: grant or deny, for the processing
of the personal data within the Rest Assured environment. The Data Protection Decision Point component is able to deliver fine-grained access control decisions, based on the individual sticky policies.

### 2.2.5.4 Authentication Component

**Owner:** 3rd party open source  
**License:** Apache License 2.0  
**Link:** [A.1.3](#)  
**Open Source Alternatives:** Any other OpenID Connect implementation:  
[https://openid.net/developers/certified/](https://openid.net/developers/certified/)  
**Purpose:**  
The Authentication component is an authentication service, using an implementation of the OpenID Connect protocol.
2.3 Building Solution Examples

2.3.1 Enabling run-time data protection assurance

Work package 5 addresses the problem of run-time data protection assurance by detecting changes in data protection risks during run time, mitigating them by appropriate adaptations, and performing other adaptations (e.g., for performance reasons) in such a way that data protection policies remain satisfied. So far, D5.1 mainly focused on the detection of data protection risks during run time. Later in the project, D5.2 and D5.3 will provide extensions ensuring the perpetuation of data protection policy compliance.

Therefore, we now focus on how to build a solution incorporating the techniques for data protection risk detection described in D5.1. For this purpose, the following steps must be made:

- **Refining the meta-model.** As explained in D5.1, Section 2.2, the proposed approach for detecting data protection risks is based on a meta-model. Both the run-time model of the cloud system and the risk patterns depend on this meta-model. D5.1, Section 3.3 describes a possible meta-model. This meta-model may have to be customized though, depending on the specific cloud system to be targeted. As an example, the meta-model in D5.1 assumes that application components are deployed in virtual machines. If the given system uses containers instead of or in addition to virtual machines, the meta-model has to be modified or extended accordingly.

- **Identification of risk patterns.** A further design-time activity that is required as preparation for the run-time data protection assurance is the elaboration of a catalog of relevant risk patterns. An initial catalog of risk patterns has been described and made publicly available under https://restassuredh2020.eu/wp-content/uploads/2018/06/Modelling-Data-Protection-Vulnerabilities-of-Cloud-Systems-using-Risk-Patterns-Technical-Report.pdf. Similarly to the meta-model, this catalog can be used as a basis but may have to be extended or modified according to the specific technologies used and their potential vulnerabilities. For identifying – and continually updating – vulnerabilities, public databases such as the Common Vulnerabilities and Exposures (CVE) database (https://cve.mitre.org/) can be used as source.

- **Implementation of monitoring adapters.** Detecting changed data protection risks during run time presupposes that the used run-time model is kept in line with the configuration of the cloud system by means of monitoring. The monitoring system described in D5.1, Section 5.3.3, handles this in a generic way and depends on specific monitoring adapters for the used systems. A monitoring adapter extracts real-time monitoring information from the monitored system and forwards that information to the monitoring gateway of the RestAssured Adaptation component. The monitoring adapter has to be implemented in a system-specific way for the used applications and infrastructure management systems, as different systems offer different interfaces or necessitate different monitoring probes to extract information about their state.

- **Handling identified risks.** The run-time data protection system of RestAssured, as described in D5.1, mainly focuses on detecting the appearance of data protection risks. When such a risk has been detected during run time, an appropriate alarm mechanism should be used to notify the system administrators and provide information about the found risk pattern to them. This can be in the form of a dashboard or by integration with the incident reporting system of the organization.

- **Optional: integration with design-time tools.** The run-time model may be initialized with a deployment plan created during the design phase with the help of some deployment planning tool. One possibility for such integration could be through a standardized language for cloud deployments, such as TOSCA (Topology and Orchestration Specification for Cloud Applications). For this purpose, the mapping between the run-time model and TOSCA, described in D5.1, Section 3.4, can be used to implement an appropriate TOSCA import interface.
In the future, the RestAssured run-time data protection will also include adaptations for ensuring continuous satisfaction of data protection requirements. For this purpose, also adaptation execution adapters will have to be implemented, similarly to the monitoring adapters mentioned above.

2.3.2 Integrating the SCANT Use Case

2.3.2.1 Background

Ami, developed and operated by Oxford Computer Consultants, is an online service in the United Kingdom that connects (i) lonely people who need help and (ii) volunteers offering help. Matching volunteers to people needing care is based on information such as the place where a person lives and their needs. These pieces of information are displayed only in obfuscated form, so as to preserve the users’ privacy. The information about people with loneliness and related needs is valuable to local authorities, who are responsible for supplying social care to persons in need within their areas.

SCANT is a tool to assist the local authorities in identifying unmet needs, whilst also preserving the privacy of the potentially vulnerable Ami users. For instance, local authorities can query with SCANT the number of Ami users with particular needs in a broad geographical region, however, individual Ami users who did not consent to the disclosure of their data will remain anonymous to the local authorities. The stored sensitive data are protected against unauthorized access. Queries from local authorities are modified automatically on the fly so that the data from Ami users who did not consent to the analytical use of their data are excluded from the results. This guarantees that local authorities never get access to data of Ami users who did not consent to this use of their data. Local authorities can still work with data of Ami users who did consent to the disclosure, and with aggregated data of Ami users who consented to aggregated usage.

2.3.2.2 Software Architecture

The software architecture of SCANT is based on the assumption that the SCANT tool itself cannot be entirely trusted. It may be hosted on a location, such as a public cloud, that provides opportunities for hostile actors to interfere. There is also the possibility of an “inside threat” amongst the staff with access to the SCANT tool at the Local Authority or Service Provider. Instead the SCANT tool delegates access to the sensitive data to the Query Gateway components. As can be seen in Figure 2.1, the SCANT tool sends its query to the Query Gateway, which uses the Data Gatekeeper to verify the data subject’s consent to the usage requested.

The SCANT tool itself is designed as an easily deployable web app, which could be rolled out on user premises. It is a python web app, hosted on a Nginx server process in a docker container. These technology choices were made to provide the following benefits.

**Python** The python language provides great flexibility while we are developing the SCANT tool. The current iteration of the tool is a prototype to demonstrate the feasibility of the RestAssured approach, rather than a finished product. The ability to make revisions quickly is a more valuable feature than the maintainability and guarantees of formal correctness of a more structured language. The availability of excellent open source Python libraries is also a valuable asset.

**Flask** Flask is a lightweight and relatively unopinionated framework for building web apps. It has proved ideal for our use case and has allowed a rapidly evolving prototype to mature as the project has progressed.

**Nginx** Nginx comes with an in-built development web server, but this is not adequate for a production system. Nginx is a performant and robust web server that handles the web request servicing and thread management for our app.
Docker  Docker provides isolation of our system from dependency on the details of the host platform. Our app is deployed as a pre-built Docker image. This allows us to assemble our system dependencies independent of the host operating system.

The SCANT code is responsible for parsing the user’s inputs, and using this to assemble a query relating to the need of interest and the region under study.

2.3.2.3 Technical Design Considerations

The SCANT application was designed around a model of an untrusted application running in a potentially insecure environment. The integrity of the system is guaranteed by the secure core of the Query Gateway, which is considered reliable. This resulted in a set of design criteria:

- SCANT cannot have access to any sensitive data.
- SCANT cannot have direct access to the user permissions.
- All sensitive data is contained within the Query Gateway.
- All access permissions were controlled by the Data Gatekeeper.

This implies:

- Query Gateway gets access permissions from Data Gatekeeper.
- SCANT acts as a user interface to the query gateway, for the purpose of querying the database, and displaying the results.
- SCANT acts as a user interface to the data gatekeeper, for the purpose of registering users and updating user consent.

Some further design constraints arose from technological concerns. Most significantly, the detailed design of the system was under development concurrently with the development of the RestAssured core components. As a result, SCANT had critical prerequisites the design of which was not stable. Flexibility and ease of applying changes was thus a key priority. To achieve this we made 3 key decisions:

1. Keep the components loosely coupled.
   This insulates SCANT from downstream changes. To a certain extent, this arises naturally from the architecture of the RestAssured system. The components communicate via web APIs, which avoids direct coupling. In addition to this the SCANT application was designed with a fairly generic query creation system that used a simplified SQL-like structure to create queries. This proved especially useful as limitations with the original Opaque implementation of the Query Gateway became apparent.

2. Develop in Python.
   Python is a lightweight scripting type language, but with expressive syntax and a powerful set of libraries. Especially relevant to our case are the Flask web server framework that allowed us to quickly produce UI for our application, and the Requests HTTP request manager library. With these components and the fairly flexible query library given above, it was fairly straightforward to adapt to emerging constraints.

3. Containerise the application to reduce dependency on host platform configuration.
   By deploying the application in its own Docker image, we avoided dealing with tiresome dependency issues on deployed host environments. This was possible as the SCANT application itself did not require exotic hosting such as SGX.

![Figure 2.2: SCANT Implementation Overview](image)

2.3.2.4 Queries

The queries generated by SCANT and processed by the Query Gateway are essentially a limited subset of SQL queries, with some attached metadata to identify the use the query is being put to and the queries it requires. A sample query is given below:
Let’s break this down a bit:

- **https://SERVER-ADDRESS/query/demo/**: This is simply the location where the Query Gateway endpoint is hosted.

- **serviceA**: This is the identity of the service using the Query Gateway. The SCANT use case was called “ServiceA” during development. This value will be given you when you register with the RestAssured service.

- **index1**: This is the index of the usage you are requesting. In this case, index1 is the use case where a local authority user is accessing the data to plan care provision in their area. (As opposed to index2, where a commercial user is planning their marketing activities with the same data.) When setting up the service with the data gateway, a separate usage is required for each distinct scenario a user may be requested to consent to.

- **data=Outcode&data=NeedID**: This section specified the column metadata: the columns that may be needed in this query. The names of the columns are highlighted in red.

- **SELECT...**: This is simply the url-escaped SQL query for:

  ```sql
  SELECT PersonID from escant where ((Outcode = 'OX2' AND NeedID = 2))
  ```

  This text is implementation dependent, but the example given shows query the list of anonymous PersonID for all consenting persons in the “OX2” postcode region (i.e. West Oxford) who have an unmet need of type 2 (i.e. “Odd Jobs”).

  Note that the columns in the where clause are those specified in the column metadata above. The Select query returns the PersonID. This may seem to violate the column metadata and anonymity constraints. However, PersonID is a special column that acts as the person-identifier. It is, by itself not sensitive data. Care must be taken to ensure that your data model does not permit indirect deanonymisation of data subjects by means of over-specific queries.

  The result of this query would be returned as a JSON object of the form:

  ```json
  [{
   "PersonID": 1,
   "PersonID": 5,
   "PersonID": 7,
   // etc...
  }]
  ```

**Query Generation** Much of the functionality of SCANT revolves around the generation of these query URLs. The SCANT program dynamically generates the queries based on the input from the UI. The UI itself is a relatively simple, consisting of a map control allow selection of a postcode region (Outcode) from a map of Oxfordshire and a social care need from a list of options.

This UI results in the selection of up to two options: Zero or one Postcode Regions, and zero or one needs. The query processor uses these to generate a tree of query terms describing the query. Unfortunately the implementation of Opaque at the time of writing was quite limited, and SQL features were not available. Most seriously, the use of the OR logical operator. This restricts the SCANT client to a maximum of 1 need and postcode region.
The tree representation records the query terms as a tree of Left-Hand-Side / Operator / Right-Hand-Side triples. Each of these triples is an example of a `QueryTerm` object. These assemble two pieces of data: 1) The query text itself (specifically the where clause) 2) The column metadata list of columns used in the query terms.

In practice, the simple queries employed by the current version of SCANT do not require this mechanism - it was created to support more general querying, and was designed before the limitations of Opaque were fully appreciated.

### 2.3.2.5 Updating the data

SCANT also contains a facility to add new records to the database, to the extent permitted by Opaque. This is handled by the Data Gateway aspect of the Query Gateway software, accessed via a web API as for queries. The URL format for this feature is:

```
http://SERVER-ADDRESS/data/v1/upload/escant
```
The data to be submitted is POSTed as a JSON object to the endpoint. For example, a SCANT object would look like:

```json
{ "City": "Burford", "FirstName": "Carlos", "Title": "Ms", "PersonId": 304, "LastName": "Franklin", "Outcode": "OX18", "IsTopNeed": 0, "NeedId": 2, "AddressLine1": "64 Frethern Close", "Need": "Odd jobs", "AddressLine2": None, "PostCode": "OX18 4NU" }
```

Once a new data subject is registered on the system by the mechanism above, their sticky policies need to be updated (by default they consent to nothing.) This is handled by the Data Gatekeeper itself, which has its own UI. The SCANT website redirects to the supplied URL in the Data Gatekeeper, using a form with the PersonID and email address of the newly registered data subject.

### 2.3.3 Integrating the Pay-As-You-Drive Use Case

#### 2.3.3.1 Background

In the past traditional car insurance policies have used basic static information about driver (age, gender, profession and previous claim history) and automobile (brand, model, year of manufacture) upon which to base premiums for insurance. The introduction of vehicle telematics has enabled a number of new *Usage-Based Insurance (UBI)* pricing models to emerge, allowing for a more fine-grained approach to risk assessment to be carried out derived from empirical driving data:

- **Pay-As-You-Drive (PAYD)** describes an automotive UBI product with a pricing model based on distance driven based on telemetric data collected while the car is being driven. The telemetric data set collected for determining the policy premium for such products is often comparatively narrow, mostly limited to odometer readings (distance travelled) and more recently in some cases including geopositioning (GPS) data sets of date, time, speed, direction and location to be used for simplistic usage analysis.

- **Pay-How-You-Drive (PHYD)** describes a more advanced UBI product with a pricing model based on driver behaviour analysis. The telemetric data set collected for determining the policy premium is a far wider telematic data set (with more resolution and accuracy) and it is processed more deeply than traditional PAYD. Not only using more finely sampled location, time of day, and distance travelled, but also taking in telematics streams of car controls (eg. steering wheel, brakes) instrumentation (tachometer) and other sensors, a driver behavioural profile can be built up algorithmically by back end analysis (such as machine learning) in cloud infrastructures.

Unless an explicit reference is made to a specific model (e.g. PHYD), this document will otherwise use the term PAYD to refer interchangeably to all the above insurance models and associated technical components and back-end cloud analysis models.
2.3.3.2 Data Model

OpenXC\(^1\), developed by Ford and Bug Labs as an open standard for accessing vehicle data, has been selected as the basis of the PAYD Data Model, as this provides direct interoperability with a growing ecosystem of both open source software and hardware for interacting with vehicle data, as well as access to advanced instrumentation data that will be explored in Phase 2 of the project.

In the case of the PAYD use case, the data model involves 3 tables:

**Person** contains sensitive personal information about the driver, including name, gender, date of birth, etc.

**Auto** contains general information about the vehicle involved in a Journey, including identifying information about the vehicle, such as its vehicle identification number (VIN).

**Event** contains telemetry that forms part of a Journey, including sensitive information such as geolocational positioning data, speed, acceleration and braking patterns, etc.

and is further elaborated in Figure 2.4:

![Figure 2.4: PAYD Conceptual Data Model](http://openxcplatform.com)

Of particular note is that while each of these tables contains different types of data, with the exception of the Person data, none of the Auto or Event data is inherently a threat to the individual’s privacy until such a time that the data is linked by a join operation across the tables. In RestAssured, not only is the sensitive data encrypted, the entire series of inner joins that create the opportunity to single out the individual through linkability or inference across the tables within a single complex query is run in an encrypted environment, with only the results explicitly consented to by the driver being returned in the query result.

\(^1\) http://openxcplatform.com
**Purpose limitation** is achieved by requiring each query to be run on a targeted service endpoint in the Query Gateway for which the purpose of data access and collection is codified as part of the requesting URI - assessed and decided upon by the Data Gatekeeper. The data subject is therefore able to pre-consent to a number of different purposes and usages, providing fine-grained control over the types and amount of data that can be accessed on a case-by-case basis. The requester then knows precisely what can be done with the data, and in which context, facilitating their compliance with the GDPR.

The individual components of the PAYD environment are elaborated in Table 2.1 below:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>openxc-vehicle-simulator</td>
<td>An application for generating simulated OpenXC vehicle trace data in real-time</td>
</tr>
<tr>
<td>openxc2event</td>
<td>A conversion utility for non-interactive conversion of OpenXC vehicle traces to RestAssured event records (JSON), for insertion into the encrypted data store</td>
</tr>
<tr>
<td>openxc2mysql</td>
<td>A conversion utility for non-interaction conversion of OpenXC vehicle traces and insertion into an unencrypted MySQL/MariaDB database</td>
</tr>
<tr>
<td>PAYD insurer client</td>
<td>The insurer-facing client application where risks to be assessed for a specific driver are selected and an adjusted risk score is provided</td>
</tr>
<tr>
<td>PAYD driver app</td>
<td>The driver-facing application that presents the latest risk and policy information to the driver, allows them to manage their consent settings, and learn more about their own driving behaviour</td>
</tr>
<tr>
<td>PAYD API gateway</td>
<td>The main application-facing API gateway that arbitrates access to multiple API/Edge service endpoints.</td>
</tr>
<tr>
<td>Consent manager</td>
<td>An end-user facing application that models consent requests, obtains fine-grained consent from the driver, and informs the creation of sticky policies</td>
</tr>
</tbody>
</table>

While the software architecture outlines the PAYD runtime environment, one aspect that has been omitted is the method in which driving data is generated and inserted into the data stores. For the purpose of the demonstration, driving data from a streaming source is pre-loaded into the Encrypted data store directly - as the RestAssured technologies mature, this will be adjusted to include periodic batch updating. In preparation for the validation, the pre-loading of the Encrypted data store is achieved by the following process:

1. The *openxc-vehicle-simulator* generates a stream of synthetic driving data in real-time
2. *openxc2mysql* intercepts this data stream and provides a local unencrypted copy of the data for interrogation by the Driver (e.g. on a trusted data store, such as the Driver’s smartphone, through the PAYD driver app)
3. *openxc2event* receives the data stream, converts it into a RestAssured event record, and submits it to the encrypted data store (e.g. in an untrusted environment, such as public cloud)

Once this is done, all access to the driving data is carried out by the respective clients through the PAYD API gateway, which communicates directly with the Query Gateway, in line with the defined high-level software architecture.
The PAYD software architecture is elaborated in Figure 2.5 below:

![PAYD Software Architecture Diagram](https://restassuredh2020.eu//)

**Figure 2.5: PAYD Software Architecture**
3 RestAssured Handbook V2

V1 of the RestAssured Handbook has focused on the components developed in the first 21 months of the project and how these have been used to build the use case solutions.

V2 of the handbook will expand this report to cover:

1. the new technical components being developed, specifically:

   (a) IBM has implemented encryption for Apache Parquet files, which is a major contribution by RestAssured to the Open Source community. Apache Parquet is a columnar storage format widely used within the Hadoop ecosystem (e.g. including Spark). Storing database information in columnar format allows for much more efficient loading of Big Data into an analytic engine, such as Spark SQL, since only those columns required for the data are required to be transferred from disk to memory. IBM has implemented encryption of Parquet files, where the decryption of the data is now done within the Spark SQL engine. This allows for Big Data files to securely reside in a public cloud, and then be utilized by a Spark SQL engine running either in a trusted cloud, or within a secure enclave, such as the AMD enclave.

   (b) Cost of security measures to be added to the SSM design time modelling tool.

   (c) Extension of domain models used by SSM to improve modelling of data flows and data lifecycle including anonymization.

   (d) Introduction of run-time risk evaluation. Developing new methods to calculate system risk levels on the fly, based upon adaptations that align with variation points in a systems design.

   (e) Thales has integrated the Data Gatekeeper with an OpenID Connect server. This allows receiving information about authenticated end-users of the Rest-Assured system, whether they are Data Consumers, Data Subjects or Service Providers.

   (f) Thales has extended the reasoning capabilities of the Data Protection Decision Point, providing a matching against a context ontology, thus providing Context Based Access Control.

   (g) Adaptant has extended the PAYD applications with support for the OAuth 2.0 protocol, which will be further extended for OpenID Connect for authenticated communication with the Date Gatekeeper by different roles.

2. The application of these technical components to the use cases. Specifically, the:

   (a) High performance computing for commercial enterprises use case will be used to illustrate authentication and authorization between components, providing end-to-end auditability across the data-lifecycle, as well as exploring the application of RestAssured technologies to Apache Spark and its related components (Shark SQL, MLlib).

   (b) PAYD use case will explore the use of adaptation to handle country-level changes, service portability through amendment of the data protection contract, and will further explore shifting away from a centralized Data Gatekeeper model in order to enable decision making and enforcement at the Edge.

   (c) IBM has implemented a version of the Pay As You Driver (PAYD) use case to illustrate the use of Apache Parquet file encryption. This allows for data files, such as those holding client and health care service provider personal information, to securely reside in a public cloud, and then be utilized by a social care application running, for example, a Spark SQL engine, either in a trusted cloud, or within a secure enclave, such as the AMD enclave.

   (d) Social care for vulnerable adults will be used to illustrate the automated risk management for run-time data protection tools.
4 Conclusions

This First RestAssured Handbook outlines our progress in developing a replicable way of applying the RestAssured technologies.

We have described, in a hand-on manner, how to use the RestAssured components through a set of implemented use cases. We have ensured a full coverage of the material required to replicate our work through a set of appendices which reference the open source libraries used and the RestAssured APIs.

As such, the handbook is well on the way to becoming an important means to achieve uptake and impact in practice.
A Appendices
A.1 References to Open Source Software

A.1.1 Parquet encryption

The changes to Parquet for the IBM-led encryption can be found in the Apache github under: [https://github.com/apache/parquet-format/tree/apache-parquet-format-2.3.1](https://github.com/apache/parquet-format/tree/apache-parquet-format-2.3.1) and in: [https://github.com/apache/parquet-mr/tree/apache-parquet-1.8.2](https://github.com/apache/parquet-mr/tree/apache-parquet-1.8.2)

A.1.2 Flask

Open source Python web server. This library is available from the Flask website: [http://flask.pocoo.org/](http://flask.pocoo.org/)

A.1.3 Keycloak

Open source implementation of the OpenID Connect protocol. The documentation can be found at: [http://www.keycloak.org](http://www.keycloak.org) The source code is available on github at: [https://github.com/keycloak/keycloak](https://github.com/keycloak/keycloak)

A.1.4 Apache Jena

Open source framework for building Linked Data applications. The documentation is available from the website: [https://jena.apache.org/](https://jena.apache.org/) The source code is available on github at: [https://github.com/apache/jena](https://github.com/apache/jena)

A.1.5 Jersey

Open Source Framework for developing RESTful Web Services in Java. The documentation and latest release are available at: [https://jersey.github.io/](https://jersey.github.io/)
A.2 References to API Documentation

A.2.1 Data Gatekeeper: Data Protection Decision Point

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.

A.2.1.1 Aggregated Authorization

- **GET** https://DATAGATEKEEPER-ADDRESS/decisionpoint/services/{serviceID}/usage/{usageID}/aggregatedAuthorization

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 serviceID and usageID path parameters, as well as the set of concerned entries in the SQL Tables as query parameters. The endpoint answers with the list of authorized entries (may be empty) associated with the registered primary identifier of the service serviceID databases. The list of concerned data must be sent as query parameters.

A.2.1.2 Specific Authorization

- **GET** https://DATAGATEKEEPER-ADDRESS/decisionpoint/services/{serviceID}/usage/{usageID}/dataSubject/{dataSubjectID}/askAuthorization

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. The endpoint answers with a signed JWT stating which data are allowed for the data subject. Thus implementing fine grained authorization. In the case where there is no authorized data, an empty json object is sent.

A.2.1.3 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.

A.2.2 Data Gatekeeper: Sticky Policy Manager

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.

A.2.2.1 Graphical Representation

- **GET** https://DATAGATEKEEPER-ADDRESS/decisionpoint/dataSubject/{dataSubjectID}/displayPolicy

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 serviceID and usageID path parameters, as well as the set of concerned entries in the SQL Tables as query parameters. The endpoint answers with the list of authorized entries (may be empty) associated with the registered primary identifier of the service serviceID databases. The list of concerned data must be sent as query parameters.
This endpoint displays a graphical representation of the security policy linked with the personal data of dataSubjectID.

### A.2.2.2 Service Registry

- **GET** [https://DATAGATEKEEPER-ADDRESS/decisionpoint/listServices](https:// DATAGATEKEEPER-ADDRESS/decisionpoint/listServices)

This endpoint lists the services registered on the Data Gatekeeper.

### A.2.2.3 Data Subject Creation

- **POST** [https://DATAGATEKEEPER-ADDRESS/decisionpoint/services/request/registerFirst](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.

### A.2.2.4 Creating Sticky Policy

- **POST** [https://DATAGATEKEEPER-ADDRESS/decisionpoint/createFirstPolicy](https://DATAGATEKEEPER-ADDRESS/decisionpoint/createFirstPolicy)

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

### A.2.2.5 Updating Sticky Policy

- **POST** [https://DATAGATEKEEPER-ADDRESS/decisionpoint/updatePolicy](https://DATAGATEKEEPER-ADDRESS/decisionpoint/updatePolicy)

This endpoint updates a Sticky Policy.

### A.2.2.6 Delete Sticky Policy

- **DELETE** [https://DATAGATEKEEPER-ADDRESS/decisionpoint/deletePolicy](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.

### A.2.2.7 Signature Sticky Policy

- **GET** [https://DATAGATEKEEPER-ADDRESS/signPolicy](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.

### A.2.2.8 Validate Sticky Policy

- **GET** [https://DATAGATEKEEPER-ADDRESS/validatePolicy](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.
A.2.3 Data Gatekeeper: Data Protection Contract Manager

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.

A.2.3.1 Display Data Protection Contract

- **GET** [https://DATAGATEKEEPER-ADDRESS/dataprotectioncontract/services/{serviceID}/displayContract](https://DATAGATEKEEPER-ADDRESS/dataprotectioncontract/services/{serviceID}/displayContract)

This endpoint displays the content of a Data Protection Contract.

A.2.3.2 Build Template Contract

- **POST** [https://DATAGATEKEEPER-ADDRESS/dataprotectioncontract/buildContract](https://DATAGATEKEEPER-ADDRESS/dataprotectioncontract/buildContract)

This endpoint gets 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.

A.2.3.3 Generate Contract

- **POST** [https://DATAGATEKEEPER-ADDRESS/dataprotectioncontract/generateContract](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.