SMARTIE

Deliverable D5.1

Integration and Validation Plan

Editor:	IHP
Dissemination level: (Confidentiality)	PU
Suggested readers:	Consortium/Experts/other reader groups
Version:	7.0
Total number of pages:	41
Keywords:	Integration and Validation Plan

Abstract

This deliverable provides complete integration and validation plan including requirements, architecture and validation and evaluation procedures.

Disclaimer

The information, documentation and figures available in this deliverable are written by the SMARTIE partners under EC co-financing (project number: 609062) and does not necessarily reflect the view of the European Commission.

Impressum

[Full project title] Secure and sMArterciTIes data management

[Short project title] SMARTIE

[Number and title of work-package] WP5Integration and validation
[Document title] Integration and validation plan
[Editor: Name, company] Peter Langendörfer, IHP

[Work-package leader: Name, company] Boris Pokric, DNET

Copyright notice

Executive Summary

According to the DoW, the purpose of this deliverable is to produce integration and validation plan taking into account outputs from WP2-WP4. This deliverable will also define the architecture of the test-beds which will be used in the validation procedure. The following activities are expected to take place in this task:

- Selection of the key project innovations coming out of work packages WP2-WP4, to be integrated and validated.
- Specification of the technical validation objectives, including suitable evaluation metrics and appropriate test plans.
- Identification of specific requirements of each validation sequence in terms of devices to be used, interfaces, protocols, legacy information systems etc.
- Integration and deployment specification and planning.

List of authors

Company	Author
UMU	Antonio Skarmeta, Rafael Marín Pérez
DNET	Aleksandra Rankov, Stevan Jokic
PTIN	Luis Cortesão, Fábio Gonçalves
NEC	Jens-Mathias Bohli, Dan Garcia, Martin Bauer
GWS	Manfred Kopielski
IHP	Peter Langendörfer, Anna Sojka-Piotrowska, Jana Krimmling

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

Abbreviations

DTLS	Datagram Transport Layer Security
IDS	Intrusion Detection System
shortECC	Short Elliptic Curve Cryptosystem
M2M	Machine to machine
PDP	Packet Data Protocol
RD	Resource Directory
CoAP	Constrained Application Protocol
XACML	eXtensible Access Control Markup Language

1 Introduction

This deliverable presents the integration and validation plan taking into account outputs from WP2-WP4. A subset of available components of the technical work packages have been selected with the focus on the importance of the components for realising practical demonstrations and deployments in WP6. The selected components are presented in a further developed view of the architecture. This will be input to WP2 to complete the architectural view. For each component, an evaluation is planned in three phases. Phase 1 considers the individual component, Phase 2 and initial communication with related components, and Phase 3 completes the integration.

The deliverable furthermore describes test-beds where the functionality can be tested within the SMARTIE platform in order to realise meaningful IoT interactions. These laboratory-type testbeds will in WP6 be extended to the demonstrator sites to be evaluated in a realistic scenario relevant to further stakeholders.

1.1 Relation to S&T Objectives

This deliverable is creating a project internal roadmap towards the integration of components. Thus it does not directly produce a relevant output of the project for the general public. However, the deliverable has a strong impact on the overall success of SMARTIE since integration is the prerequisite for successful deployment of demonstrators, in particular toward achieving Objective O5 “Demonstrate the project results in real use cases.”

1.2 Beyond State-of-the-Art Contributions

This deliverable is not discussing new features or scientific advances but it is merely kind of a refined working program to ensure proper and timely integration of SMARTIE components. The deliverable itself does not contribute to the state of the art and also not beyond state of the art. But if the integration steps planned in this deliverable will be accomplished successfully the partly integration of innovative SMARTIE components in an IoT platform will go beyond state of the art.

2 Overview

2.1 Purpose and Scope of the WP5

The purpose of WP5 is to integrate key innovation components for secure data exchange and storage developed within WP2 to WP4 within a single platform and to validate such new complete system on several different experimental test-beds enabling three different smart city test scenarios.

2.2 Purpose and Scope of Task T5.1

This task aims to produce an integration and validation plan for each of three planned test scenarios with the aim to validate the platform performance, the functions provided by every component as well as the interaction between the user and the IoT devices and the new platform.

The following activities are expected to take place in this task:

- Selection of the key project innovations coming out of work packages WP2-WP4, to be integrated and validated.
- Specification of the technical validation objectives, including suitable evaluation metrics and appropriate test plans.
- Identification of specific requirements of each validation sequence in terms of devices to be used, interfaces, protocols, legacy information systems etc.
- Integration and deployment specification and planning.

2.3 Purpose and Scope of the Document

It is the purpose of this document to provide the complete integration and validation plan of the newly integrated SMARTIE platform including:

- Requirements for each component integrated
- Architecture of the new platform with all relevant APIs implemented
- Validation and evaluation procedures.

2.4 Structure of the Document

The rest of this deliverable is structured into 3 sections. Section 3 deals with the integration itself first describing the approach and then providing detailed information on how individual components will be integrated including a timeline. Section 4 discusses the validation scenarios that will be used to assess the integration done. The document closes with a short conclusion.

3 Integration of Components and System Validation

3.1 Approach to integration and validation

The innovative SMARTIE components have been developed in WP3 and WP4. A first architecture of a platform integrating these components, together with possible interactions between the components has been described in D2.3. WP5 will integrate the relevant components required for the testbeds. The integrated components will be available to the demonstrator sites and support the realisation of an IoT system that will enable sharing large volumes of heterogeneous information for use in smart city applications ensuring security in data exchange and storage as well as the trust in information delivery and the user's privacy.

The SmartData platform provided by PTIN builds the basis for the communication and integration of many components. However, the SMARTIE platform as a whole is a wider concept and also integrates components that do not depend on the SmartData platform.

The system performance and security will be evaluated and validated through realisation of three selected scenarios as described in D2.1 that will be realised in WP6. To successfully realise this integration it is necessary to conduct validation tests through the process. We define therefore the following phases, where individual components, limited APIs between selected components, and full IoT interactions are tested:

Phase 1: Basic unit tests

This refers to the basic components functionality test.

Phase 2: Inter- component test

This specifically refers to the 'horizontal' interaction between different components within the SMARTIE layer, to test standard protocols and APIs.

Phase 3: Complete system test

This refers to the vertical interaction realising an IoT interaction. It involves the new SMARTIE components and applications on one side as well as the OS (network protocols, File System integration, memory management etc) and PTIN platform on the other side (AA server, APIs. etc).

The SMARTIE project has as the main focus to protect the security and privacy of IoT systems and data when it is shared in platforms that are deployed in the cloud. While security and privacy are essential for the acceptance of the IoT, it is in the nature of a large IoT system that it must be performant and scalable. Therefore our validation metrics are focused on those three aspects:

1. **Security** is the major issue to be addressed due to the nature of the project. Not only is security required to protect the privacy of the users, but it is also crucial to ensure the access control to the resources and the verification of the data provided by devices. The nature of the presented solution requires the guarantee of security, privacy, integrity and confidentiality. Specifically the issues considered are:
    - Authentication
    - Integrity
    - Freshness
    - Privacy/Confidentiality.
2. **Performance** is a relevant aspect due to the low computation capacity of constrained devices and the small bandwidth of limited wireless networks.
    - Protocol implementation size for devices
    - Payload length for wireless network

- Reliability to provide good delivery rates in end-to-end communications.
- Latency time in the end-to-end transmission communication.

3. **Scalability** in IoT applications refers to a number of IoT devices that can be connected to the platform and it applies to:
   - Number of resources and services that can be managed by the Smartdata platform.
   - Distribution of security mechanisms to control the global access of IoT devices through Internet.

The following tables list more specific preconditions and tests for the three phases per component.

3.1 Test cases

3.1.1 DCapBAC

Phase 1:	**Basic unit tests (component level)**
TC name	DCapBAC – Capability Manager
Preconditions	Install and start DCapBAC Capability Manager component
Execution	Capability Token request and generation
Expected results	Inspect results to check that Capability Manager receives a capability request and generates a new token with the resource access.
Expected completion	M24

Phase 2:	**Inter- component test (** SMARTIE components)
TC name	DCapBAC with COAP+DTLS
Preconditions	Integration of DCapBAC with COAP+DTLS
Execution	Perform COAP+DTLS exchange for Capability token requests and generations.
Expected results	Inspect results to check that a user requests a new token to access a resource and the Capability manager responds a new capability token for this resource.
Expected completion	M24

Phase 3:	**Complete system test**
TC name	Integrated DCapBAC in Smartdata platform
Preconditions	Install and start DCapBAC component, check mutual accessibility between Smartdata platform and DCapBAC component
Execution	Perform token requests and generations from the Smartdata platform.
Expected results	Inspect results to check new token is requested to the Smartdata platform and the Capability manager responds a new capability token.
Expected completion	M30

3.1.2 XACML

Phase 1:	**Basic unit tests (component level)**
TC name	XACML – Policy Decision Point
Preconditions	Install and start XACML component
Execution	Perform XACML queries and decisions
Expected results	Inspect results to check that XACML receives authorization queries and responds permit/deny decisions.
Expected completion	M24

Phase 2:	**Inter- component test (** SMARTIE components)
TC name	XACML and DCapBAC
Preconditions	Integration of XACML with DCapBAC
Execution	Perform resources access request for XACML user authorization and DCapBAC token generation.
Expected results	Inspect results to check that a user requests a resource access, the XACML evaluates the user credentials and take the permit/deny decision for the Capability manager to generates or not a capability token for the resource access.
Expected completion	M24

Phase 3:	**Complete system test**
TC name	Integrated DCapBAC in Smartdata platform
Preconditions	Install and start DCapBAC component, check mutual accessibility between Smartdata platform and DCapBAC component
Execution	Perform token requests and generations from the Smartdata platform.
Expected results	Inspect results to check resource access is requested to the Smartdata platform and the XACML checks the user authorization to indicate if the DCapBAC generates a capability token.
Expected completion	M30

3.1.3 shortECC

Phase 1:	**Basic unit tests (component level)**
TC name	shortECC
Preconditions	Implementation of shortECC in C and Java libraries
Execution	shortECC signing and verification
Expected results	Inspect results to check that a PC generates a shortECC sign with Java library and an embedded device verifies the correct sign by the C library.
Expected completion	M24

Phase 2:	**Inter- component test (** SMARTIE components)
TC name	shortECC integrated in DCapBAC
Preconditions	Integration of shortECC and DCapBAC components
Execution	Token generation and validation with shortECC
Expected results	Successful verification for correct shortECC signing
Expected completion	M27

Phase 3:	**Complete system test**
TC name	Integrated shortECC and DCapBAC within Smartdata platform
Preconditions	Install and start shortECC and DCapBAC in SMARTIE platform
Execution	Token generation and validation with shortECC through Smartdata platform
Expected results	Inspect results to check resource access is requested to the Smartdata platform and the DCapBAC generates a capability token with shortECC signing. The embedded device verifies the shortECC signing before to permit the access for the resource.
Expected completion	M30

3.1.4 IDS

Phase 1:	**Basic unit tests (component level)**
TC name	IDS
Preconditions	Implementation of IDS in C and Java libraries
Execution	IDS training and detection
Expected results	Training: investigates training data to build up knowledge base / history for detection, Detection: use knowledge base / history for detection of anomalous data
Expected completion	M24

Phase 2:	**Inter- component test (**SMARTIE components)
TC name	IDS integration
Preconditions	Integrate and run IDS on SMARTIE node
Execution	Training and detection running on SMARTIE node
Expected results	Successful build of knowledge base / history; detection runs with generated knowledge base / history
Expected completion	M27

Phase 3:	**Complete system test**
TC name	IDS tuning
Preconditions	IDS parameter tuning for application on SMARTIE platform
Execution	Training and detection SMARTIE platform with SMARTIE traffic data
Expected results	Use SMARTIE traffic for training and detection; build up knowledge base / history and successfully detect and report intentionally created anomalies.
Expected completion	M30

3.1.5 CP-ABE

Phase 1:	**Basic unit tests (component level)**
TC name	CP-ABE
Preconditions	Install and start CP-ABE component
Execution	Perform CP-ABE ciphering and deciphering
Expected results	Inspect results to check that information is ciphered and deciphered by CP-ABE schema.
Expected completion	M24

Phase 2:	**Inter- component test (** SMARTIE components)
TC name	CP-ABE with COAP+DTLS
Preconditions	Integration of CP-ABE with COAP+DTLS
Execution	Perform COAP+DTLS exchange for CP-ABE ciphering and deciphering.
Expected results	Inspect results to check that a data message is ciphered and sent to other machine that can decipher the information in the message.
Expected completion	M30

Phase 3:	**Complete system test**
TC name	Integrated CP-ABE in Smartdata platform
Preconditions	Install and start CP-ABE component, check mutual accessibility between Smartdata platform and CP-ABE
Execution	Perform CP-ABE ciphering and deciphering by the platform and users
Expected results	Inspect results to check that the platform cipher a data message to send the user that is able to decipher the data with his CP-ABE key. The content of the messages must be made legible only to authorized users that know the CP-ABE key.
Expected completion	M30

3.1.6 Distributed Kerberos/Capability tokens/Actuation

Phase 1:	**Basic unit tests (component level)**
TC name	Distributed Kerberos Server
Preconditions	none
Execution	The servers use secure multi-party computation, which is a complex cryptographic operation. This test will analyse the practicality of the distributed Kerberos protocol. The test will use commodity PCs as a platform for the two Servers. In order to measure the throughput, the clients will send ticket requests at an increasing rate until a saturation of the servers is reached.
Expected results	The result allows determining the performance and a price for distributed Kerberos authentication. It is expected that the costs are higher than standard Kerberos, but tolerable considering the additional security and applicability gained.
Expected completion	M22

Phase 2:	**Inter- component test (** SMARTIE components)
TC name	Kerberos with authentication Tokens
Preconditions	Kerberos Server and Capability tokens available.
Execution	In this test, the Kerberos servers will not only issue a Kerberos token, but also include a capability token. A sensor that can process the token will be present. In the test, a client requests a token for accessing a certain device. The server authenticates the client and check if the access is authorized. In this case, the servers generate a joint Kerberos and capability token and send it to the client. The client sends the request to the device. In the device, the tokens are validated and an encrypted response is sent to the client. The test measures the throughput of the sensor when Kerberos/capability tokens are processed by sending requests at an increasing rate to the sensor until saturation is reached. A second test, will measure the total time of the token and data requests.
Expected results	It is expected that the token verification takes no significant overhead compared to state-of-the-art security solutions. The total process is expected to be <5 sec so that it is practical for most interactions.
Expected completion	M24

Phase 3:	**Complete system test**
TC name	Actuation using a remote ticket granting service
Preconditions	Kerberos server is available via the Internet, Kerberos-enabled device is installed to control a relay for opening a door.
Execution	A user will request a Kerberos token with capability token from a remote server in the platform. The user will build up a local connection to a device in the demonstrator site to perform an actuation. The messages in the processing flow will be analysed to verify secure communication.
Expected results	The request for the token followed by the contact to the service is transparent for the user. Delay will be tolerable for secure operation. The communication is secure against attacks in the network.
Expected completion	M26

3.1.7 PrivLoc

Phase 1:	**Basic unit tests (component level)**
TC name	PrivLoc
Preconditions	The unit test is running in the scenario of a Geo-Fencing application. A Geo-Fencing application is composed of i) subscribers, who register a geometric rectangle area (i.e., fence) in the application; and ii) publishers, who update their location coordinates to the application so that the subscribers of an interfered subscribed fence would be notified.
Execution	For correctness test, we will observe that the output encrypted locations are fully obfuscated, as well as the result of the spatial queries on the encrypted locations such as CONTAIN and CROSS. We will also evaluate the performance of PrivLoc w.r.t the spatial databases (e.g. MySQL, PostgreSQL) in the Geo-Fencing application. The database will be already populated with a certain number of subscriptions. We will simulate mobile objects by generating random location update of their routes to the system. By increasing the number of clients (mobile objects), we measure the average response time of a query and the maximum throughput (i.e., total number of processed queries at a unit time) of the system.
Expected results	We expect that the spatial query result on the encrypted locations will be the same as that on the original location data. As to performance test, notice that PrivLoc only acts as a proxy and it still uses underlying spatial database to process queries. The evaluation is to show that the encryption process of the coordinates will not make PrivLoc a bottleneck in the system. In fact, it should have a much better performance so that the service providers would prefer to host a light-weight proxy in their corporation environment, and outsource the database (storage and searching functionality) to the cloud, while the user privacy is still preserved. Therefore, we expect that PrivLoc has a bigger peak throughput of publish queries than the databases.
Expected completion	M24

Phase 2:	**Inter- component test (** SMARTIE components)
TC name	PrivLoc as an enabler in the SmartData platform
Preconditions	Privloc implementation accessible as an enabler.
Execution	The location data from SmartData platform will serve as the input to PrivLoc proxy. We will test the performance of running PrivLoc as the middle layer between the SmartData platform and the outsourced spatial database. We will measure the average delay of a location query latency in the system comparing to the system without PrivLoc.
Expected results	We expect that the delay of a location query would be less than 10% of the original query latency. Moreover, when the size of the database increases (i.e., more subscriptions in the database), the delay of a query will decrease.
Expected completion	M26

Phase 3:	**Complete system test**
TC name	Vehicle Monitoring
Preconditions	GPS data of vehicles available in the SmartData Platform.
Execution	Vehicles moving around in the city encrypt their location data and regularly upload their

	movement to the SmartData platform. Clients, e.g. city officials, have subscribed to zones of interest and in the following receive reports about the vehicles entering and leaving the area. The test validates the practicability of the use of privloc for location monitoring.
Expected results	No significant overhead incurred for the moving device and the client.
Expected completion	M28

3.1.8 RD

Phase 1:	**Basic unit tests (component level)**
TC name	**RD**
Preconditions	Install and start RD component
Execution	Perform resource registration/update and delete
Expected results	Inspect results to check that a new resource is added to the RD; changes are included with an update and resources removed with delete
Expected completion	M24

Phase 2:	**Inter- component test (** SMARTIE components)
TC name	RD with DCapBAC
Preconditions	Install and start RD and DCapBAC components and check mutual accessibility
Execution	Perform login to the RD
Expected results	Success/failed login for correct/incorrect credentials
Expected completion	M25

Phase 3:	**Complete system test**
TC name	Integrated RD
Preconditions	Install and start RD component, check mutual accessibility between SMARTIE platform and RD
Execution	Perform resource adding by platform/update/delete
Expected results	Inspect results to check a new resource is added to the RD, specified content changed by update and removed by delete
Expected completion	M28

3.1.9 RD with secure storage

Phase 1:	**Basic unit tests (component level)**
TC name	RD with secure storage
Preconditions	Install and start RD component
Execution	Register resource with encryption request
Expected results	Test access to encrypted resource using correct and incorrect credentials
Expected completion	M24

3.1.10 Processing Flow Optimization

Phase 2:	**Inter- component test (** SMARTIE components)
TC name	Processing Flow Optimization and Capability Management
Preconditions	Capability Management and Processing Flow Optimization installed
Execution	Perform token request to Capability Management and access Processing Flow Optimization with Token
Expected results	Processing Flow Optimization checks token and allows execution or not.
Expected completion	M28

Phase 2:	**Inter- component test (** SMARTIE components)
TC name	Processing Flow Optimization and Resource Directory / *Network Information Directory*
Preconditions	Resource Directory / *Network Information Directory* and Processing Flow Optimization installed + Capability Management
Execution	Perform request to processing flow optimization and check if the required information from Resource Directory / *Network Information Directory* can be retrieved using Token provided by user
Expected results	Processing Flow Optimization can retrieve information from Resource Directory / *Network Information Directory* that is the basis for finding the optimal processing flow
Expected completion	M29

Phase 3:	**Complete system test**
TC name	Complete Processing Flow Optimization
Preconditions	Resource Directory / *Network Information Directory* and Processing Flow Optimization installed + Capability Management
Execution	Perform request to processing flow optimization that results in mapping the abstract processing task plan to the sensing and processing flow resources according to the optimization function
Expected	Optimized processing flow

results	
Expected completion	M30

4 Integration and validation plan

Figure 1 depicts our view of the SMARTIE platform regarding the integration plans with the components that will be made available for the demonstrators. In the following we present the IoT Interactions that can be realised using the integrated components. These plans are then used to derive the priorities and timing for integration and validation within the three phases for the individual components.

4.1 Integration scenarios

The following sections describe several IoT Interactions that involve multiple components in the SMARTIE platform.

4.1.1 Validation Testbed for Capability Manager within the SMARTIE Platform

This subsection defines a testbed for the validation of the control access from users towards the sensors through the SmartData platform in order to request information and actuation. In this testbed the integration of Capability Manager, Policy Decision Point in the SmartData platform will be validated. Moreover, COAP and DTLS transportation will be validated.

1. The user sends a request to the Capability Manager to retrieve a capability token. This request contains the target sensor as well as the action that is intended to be performed. As an example, the user is requesting retrieving the temperature of a sensor.
2. The user initiates a DTLS exchange for securing the request with the Capability Manager.
3. The Capability Manger then authenticates the user.
4. If successful, The Capability Manger sends a XACML Request to the PDP for authorization for the action the user asks for.
5. The PDP evaluates the authorization according to the XACML policies and the credentials of the user.
6. The Capability Manager then receives a XACML Response saying if the request is Permitted or Denied.
7. When the action is permitted, the Capability Manager generates a token with the action requested and the shortECC signature.
8. The Capability Manager sends the generated tokens to the user. In other case, the user receives an error message for the denied authorization.
9. It shows an invalid access to sensor where the user sends a query without the capability tokens.
10. The sensor will deny any access to information or actuation over its resources.
11. To establish a secure M2M communication, the user initiates a DTLS exchange with the sensor.
12. Then, the user can send a secure query containing the capability token.
13. The sensor validates the shortECC signature of the token and checks the requested resource.
14. If the signature and resource are valid, the sensor sends back a response to the user with the requested information or actuation.

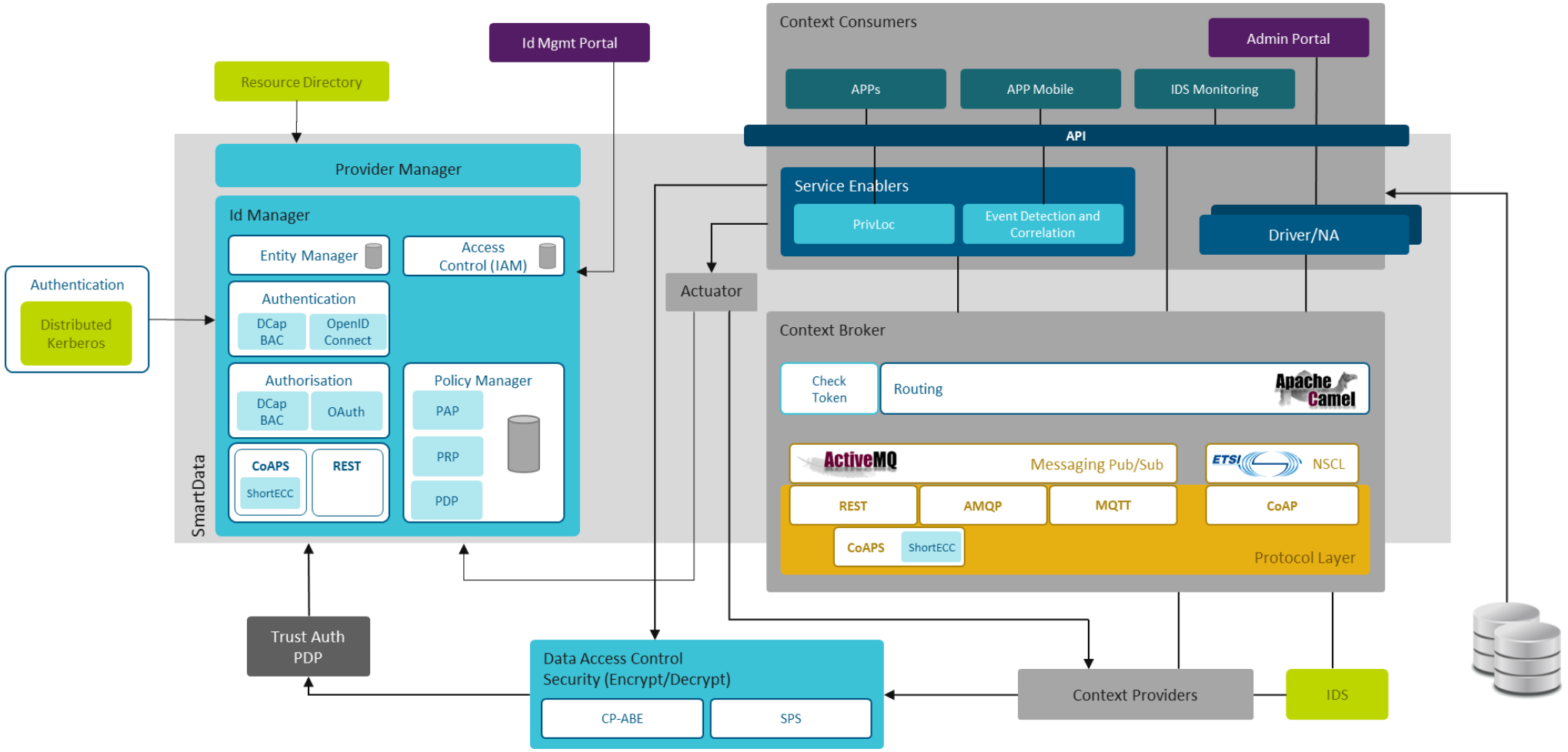

Figure 1: Smartie Platform with all components considered for integration

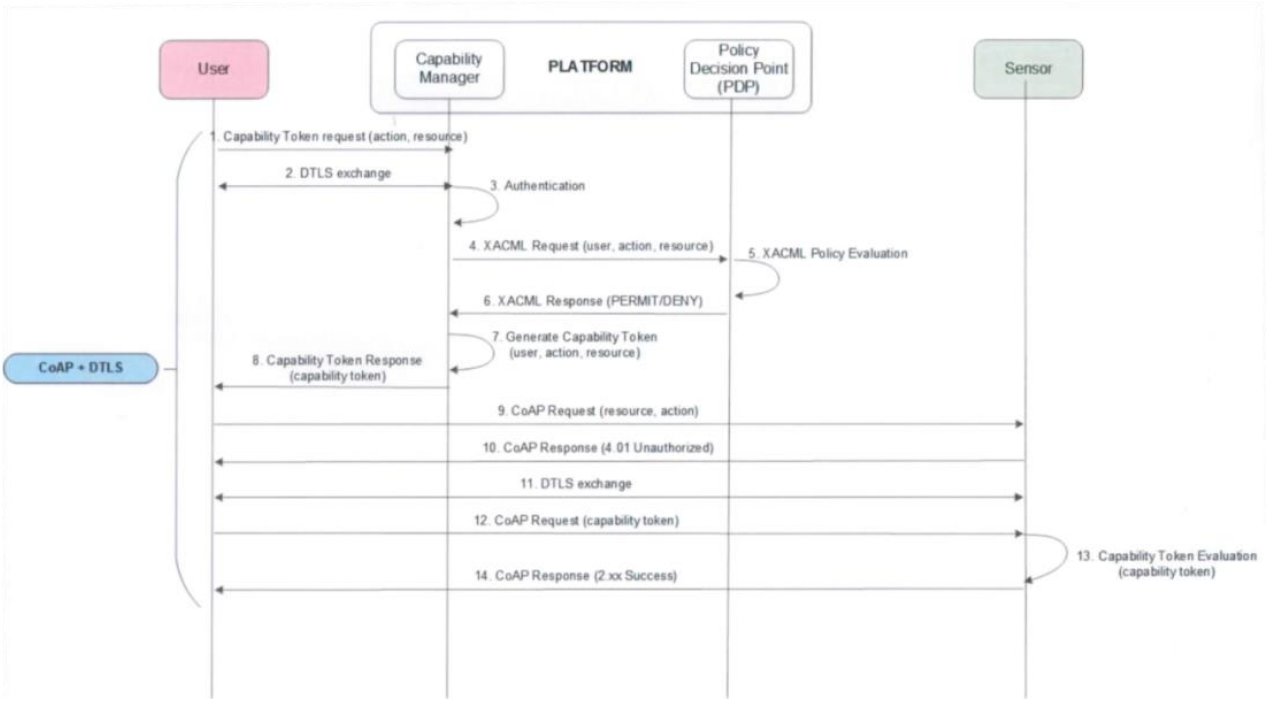

Figure 2: Testbed for getting a capabilitxy token

4.1.2 Validation Testbed for Request/Response using Kerberos

This subsection describes a testbed for testing the integration of Kerberos with Sensors and Clients in a scenario, where a client retrieves information from a sensor or starts and actuation. We assume that the discovery of the device offering the service the user is interested in, is already done. Also we can assume that the user in each scenario did not perform a previous authentication with the Authentication Server (AS) of Kerberos (KRB).

1. The user authenticates with the AS and receives a Ticket Granting Ticket (TGT).
2. The user upon successful authentication receives a TGT.
3. With the TGT the user is able to require a Service Ticket (ST) to access a service of a mote.
4. The Ticket Granting Server (TGS) will analyse the request, and send a XACML Request with the actions the user intends to perform.
5. The PDP will resolve the request and return a Permit or Deny to the TGS.
6. The TGS if the PDP responses positively, will generate a Ticket with a Capability token.
7. Along with the Ticket, a capability token as authorization will be embedded into the KRB ticket.
8. Now the user is able to send a ticket to the Mote containing the Request to the service.
9. The Mote will respond with the result of checking the Ticket and the request, checking the Capability Token is valid and matches the actual request of the user.

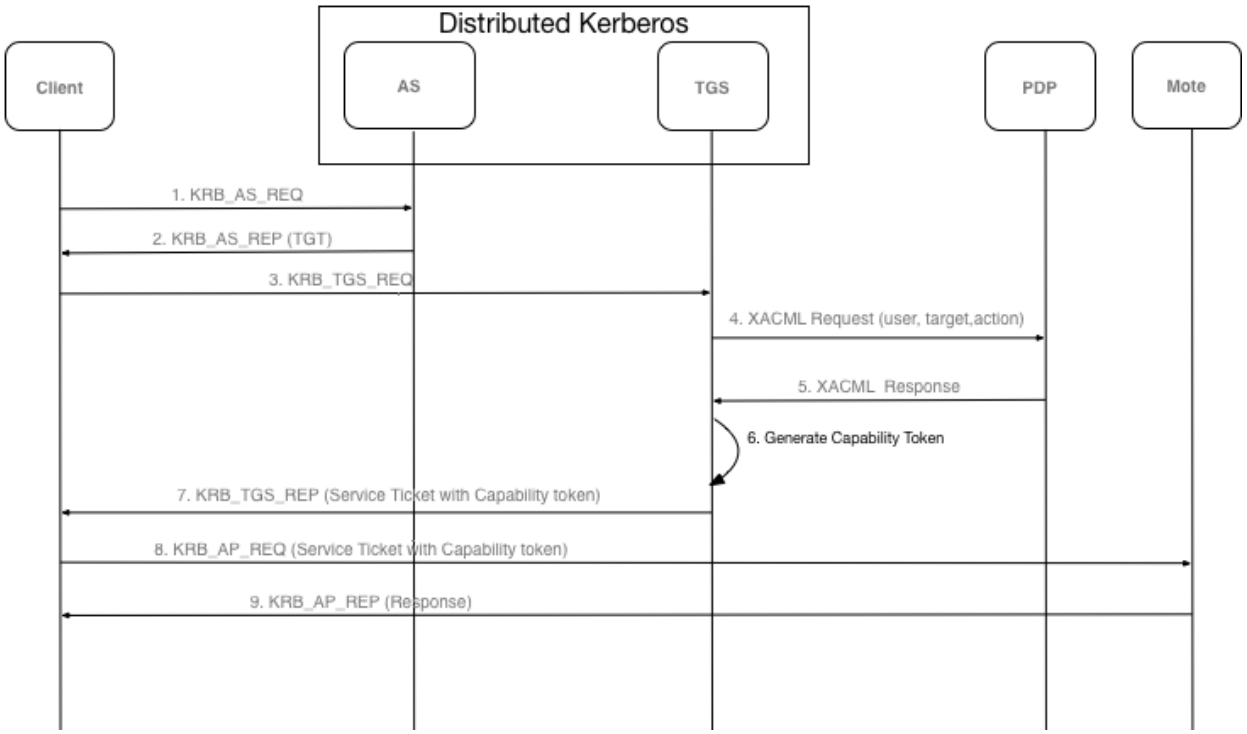

Figure 3: Testbed for requesting and using a Kerberos ticket with capbility token

4.1.3 Validation Testbed for Subscription

This subsection defines a testbed for the validation of the subscription from users to the SmartData platform in order to access the published data from the embedded devices. In this testbed, the integration of Topic Manager and CP-ABE in the SmartData platform will be validated.

1. The Topic Manager requests the generation of tokens in the Capability Manager and sends the subscription to sensors' topics with the capability tokens. So, the Topic Manager will receive every new topic from the sensors. This step contains the same communication exchanges as the previous testbed.
2. A user is able to subscribe to a topic within the Topic Manager.
3. A DTLS connection is established with the Topic Manager.
4. The Topic Manager authenticates the user through the PDP.
5. The PDP checks the authentication of the user to perform the requested subscription.
6. The PDP responds a decision of PERMIT or DENY
7. The Topic Manager sends the decision to the user.
8. The User receives the successful subscription or not.
9. Each time the sensor generates new information, the Topic Manager will send the data ciphered with CP-ABE to the user.
10. The user will decipher the received information with its keys.

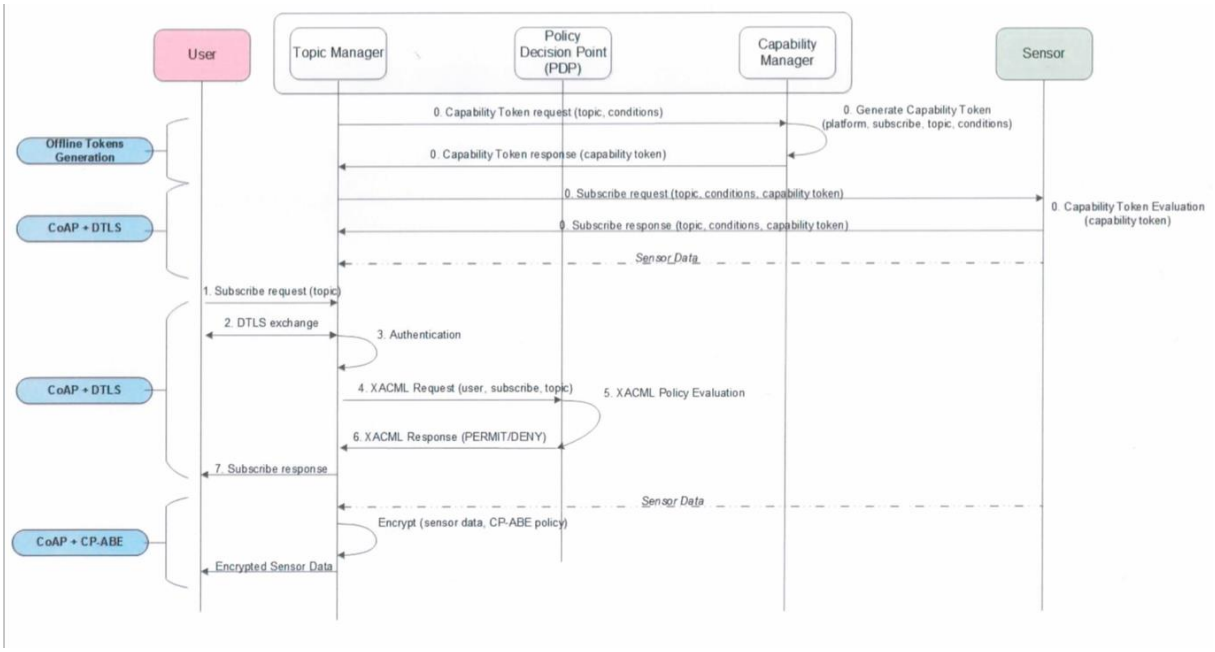

Figure 4: Testbed for Smart Building Subscription

4.1.4 Validation Testbed for Subscription using Kerberos

This subsection defines a testbed for a variant using only standard block ciphers. In this testbed, the integration of Kerberos and CP-ABE in the SmartData platform in a subscription scenario will be validated.

1. The user authenticates with the AS and receives a Ticket Granting Ticket (TGT).
2. The user upon successful authentication receives a TGT.
3. With the TGT the user is able to require a Service Ticket (ST) to access a service of a mote.
4. The Ticket Granting Server (TGS) will analyse the request, and send a XACML Request with the actions the user intends to perform.
5. The PDP will resolve the request and return a Permit or Deny to the TGS.
6. The TGS if the PDP responses positively, will generate a Ticket with a Capability token.
7. Along with the Ticket, a capability token as authorization will be embedded into the KRB ticket.
8. Now the user is able to send a ticket to the Mote containing the Request to the service.
9. The Mote will respond with the result of the result of checking the Ticket and the request, checking the Capability Token is valid and matches the actual request of the user.
10. As soon as new data is generated that matches the subscription of the user, the mote will send that information, secured with the session key, and identifier of the session.

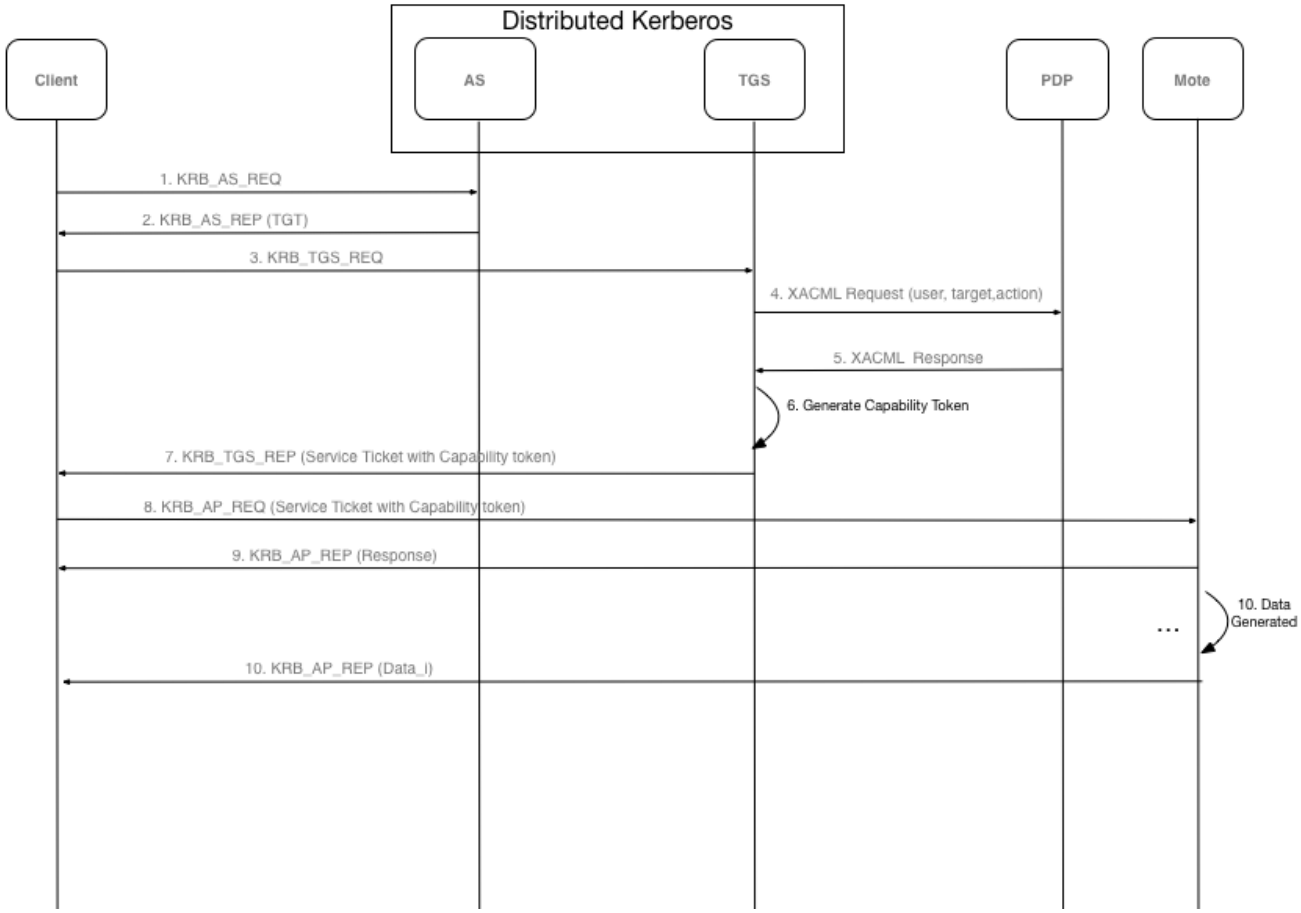

Figure 5: Testbed for Smart Building Subscription with Kerberos

4.1.5 Validation Testbed for GeoFencing ServicePrivLoc

This subsection defines a testbed for the validation of the privacy-preserving geofencing service offered through the SmartData platform. This scenario includes the deployment of a new service enabler in the platform that manages subscriptions and performs the geofencing operation on scrambled location data.

1. The user connects to the geofencing enabler to subscribe to an area of interest.
2. Moving objects measure their position using GPS sensors.
3. The object encrypts the location and sends the encrypted location update to a Smartdata platform topic.
4. The geofencing enabler subscribes to a Smartdata platform topic to process incoming location updates.
5. The geofencing enabler generates a notification message to the user, if a moving object crosses the border of one of the areas of interest.
6. The user can delete his subscription to an area of interest or add further subscriptions.

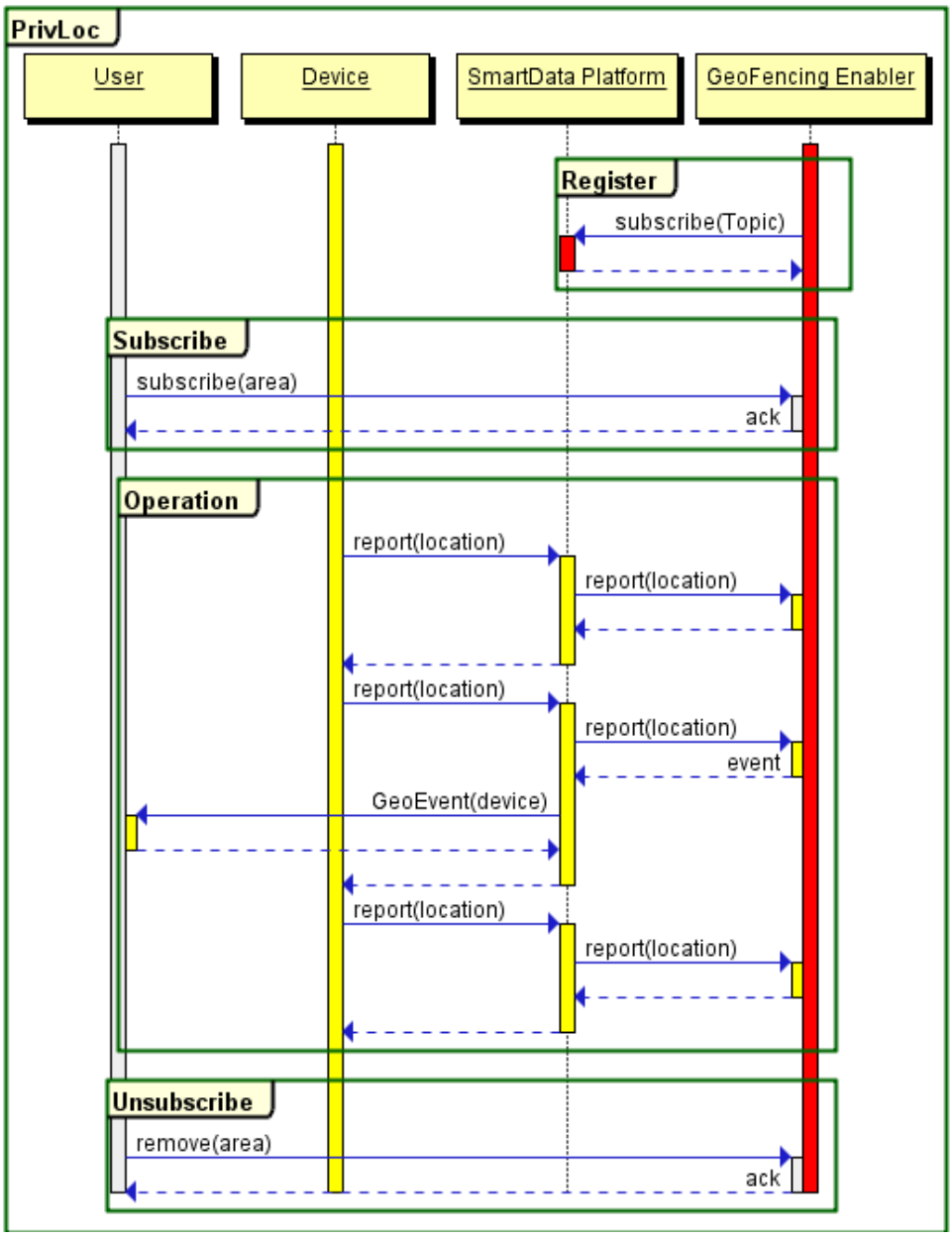

Figure 6: Interaction flow for Geofencing

4.1.6 Validation Testbed for Device Registration in Smart Public Transport

Here a testbed for validation of the IoT device (sensor) registration process to the smart data platform is described that will enable to supply the data to the resource end point, i.e. Virtual Resource as well as the secure storage in a RD database.

All IoT devices need to register to RD. The diagram below illustrates the secure registration of the environmental monitoring device (EB800) to the RD.

To realise this it is necessary to successfully integrate two new components:

- Distributed Kerberos
- RD with secure storage

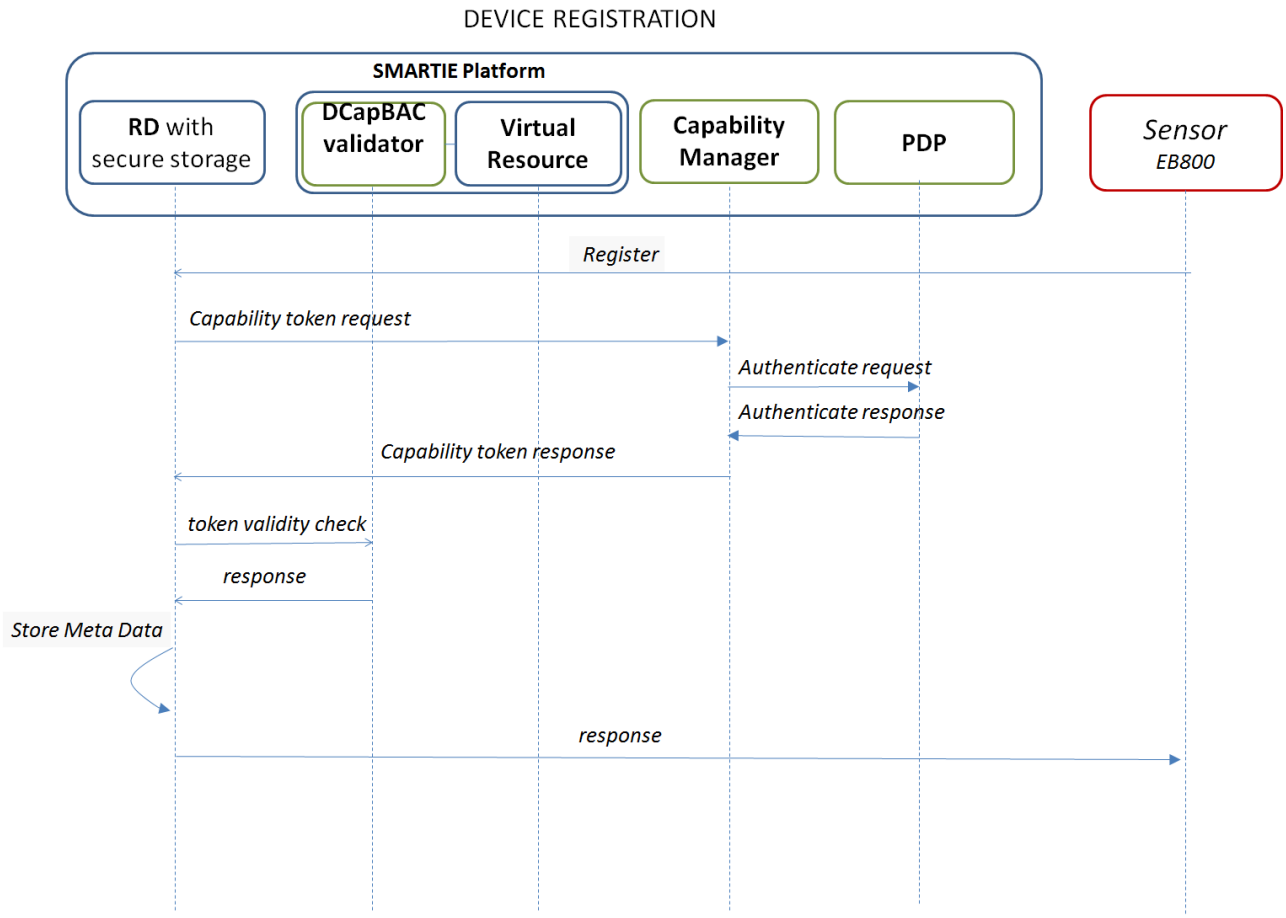

Figure 7: Device Registration

4.1.7 Validation Testbed for Data Upload and Retrieval in Smart Public Transport

The testbed described here is used to for validation of the Data Upload from IoT device to the SmartData platform as well as Data Retrieval by the users from the SmartData platform. The validation is focused on integration of Capability Manager (with DCapBAC), secureCoAP (CoAP+DTLS) , Token Validator (DCapBAC) and Virtual Resource Unit.

1. A sensor (e.g. EB800 for monitoring environmental parameters) registered to the SmartData platform uploads data in set intervals to the Virtual Resource using http and sCoap protocols. Fleet management devices transfer data to the back-end infrastructure using the same protocols.

2. User authentication to the system: A user sends an authentication (token) request to the Distributed Kerberos component integrated in the platform and receives the response/token back.

3. A user requests the list of resources from the resource directory using the authentication tokens and receives the list of sensors and services upon token validation.

4. A user then requests the token from the capability manager in order to get access to data via a web service uploaded by registered devices. Access to the environment (pollution) data can be performed using geographical location.

5. Data is transferred to the user via DTLS data transfer.

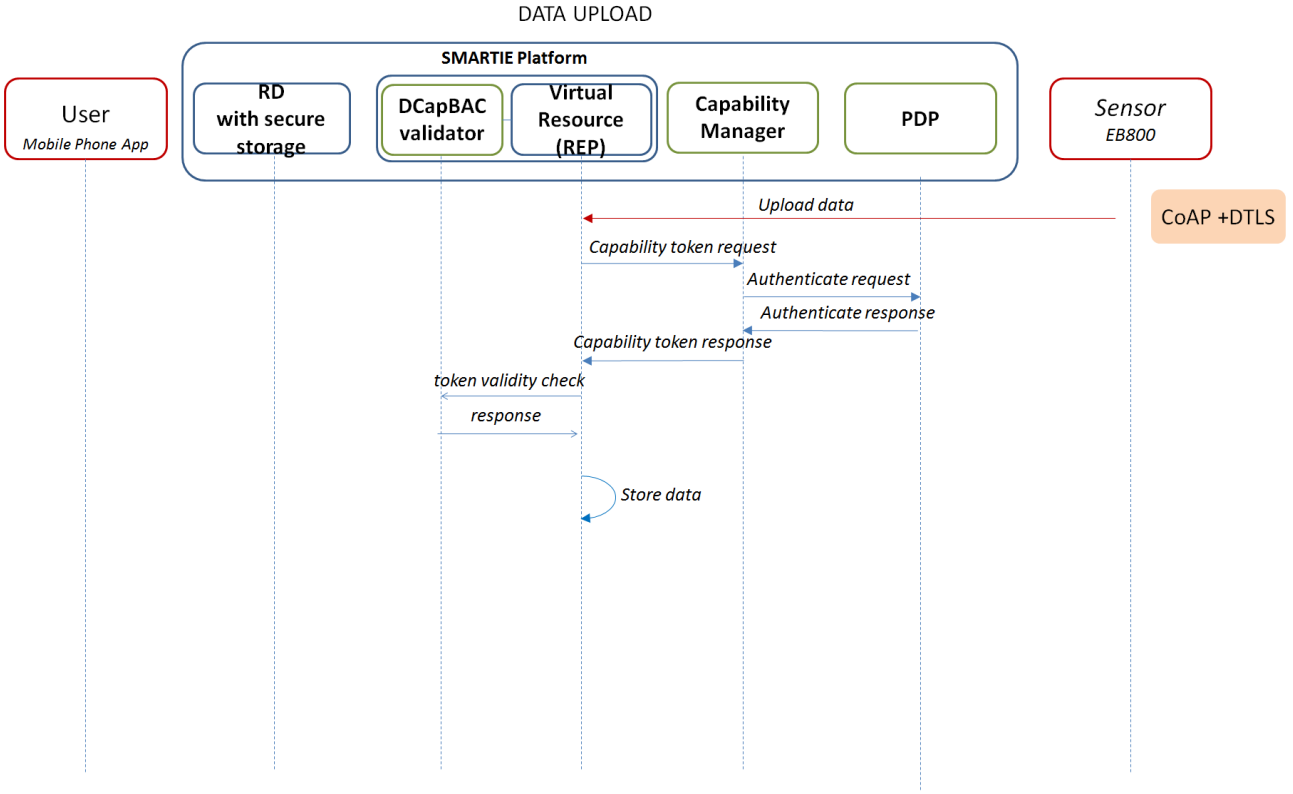

Figure 8: Data Upload

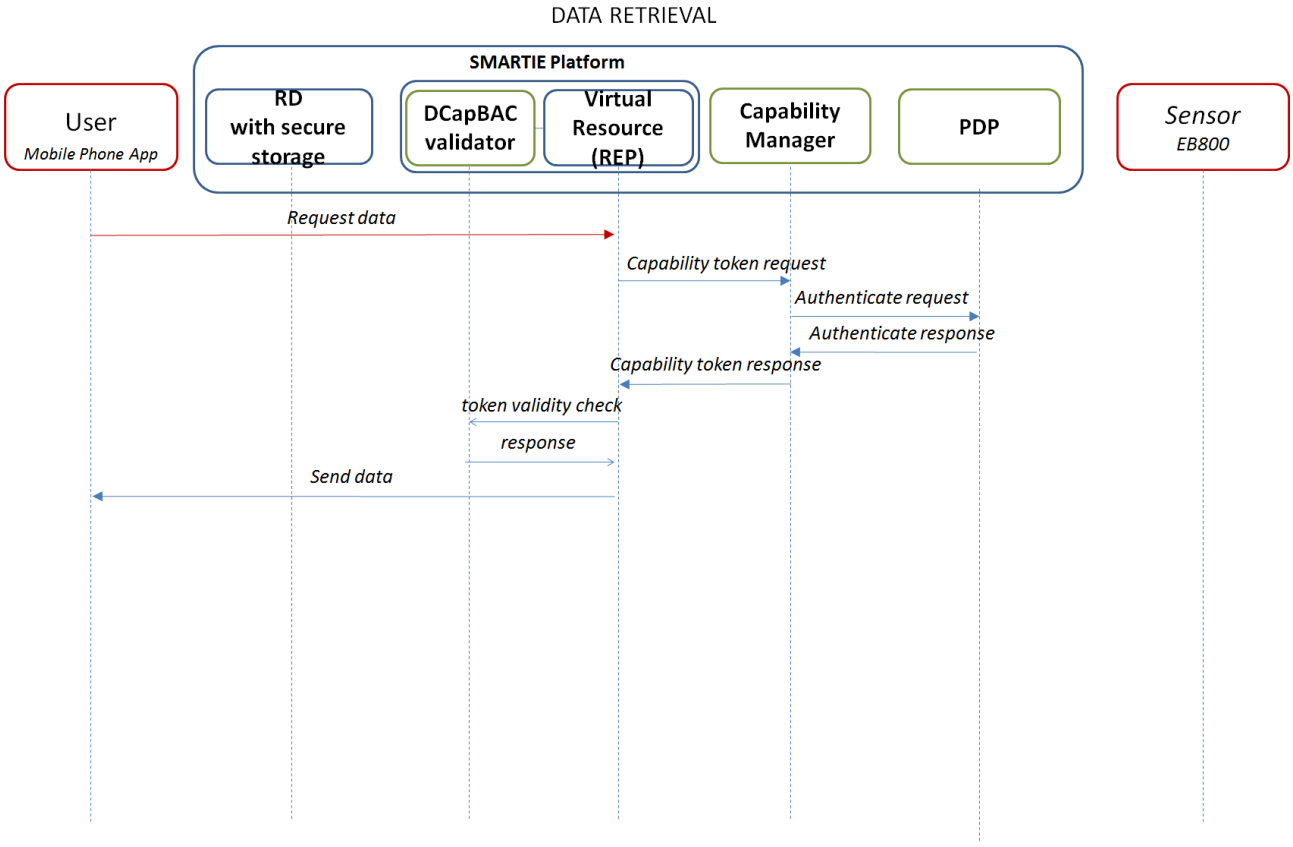

Figure 9: Data Retrieval

4.1.8 Validation Testbed for data upload & retrieval in Smart Traffic Control

This subsection defines a testbed for registration of devices and upload data to the SmartData platform. In this testbed, the integration of Capability Manager and Resource Directory in the SmartData platform, as well as the implementation of the TS Wrapper will be validated.

1. The device connects to Traffic Safe Central Unit via TLSoverIP.
2. The TS Wrapper requests a token from the Capability Manager to access a virtual entity representing the device in the smart data platform.
3. The Capability Manager authenticates the request, generates a signed token to access the virtual entity and sends it to the TS Wrapper.
4. TS Wrapper registers the device in RD.
5. TS Wrapper sends data to the platform periodically.
6. The user sends a request to Capability Manager via web application to access data from virtual entity.
7. The Capability Manager authenticates the request and sends access token to user.
8. The user requests data from virtual entity with capability token.
9. If the token is valid, the virtual entity sends the requested data back to the user.

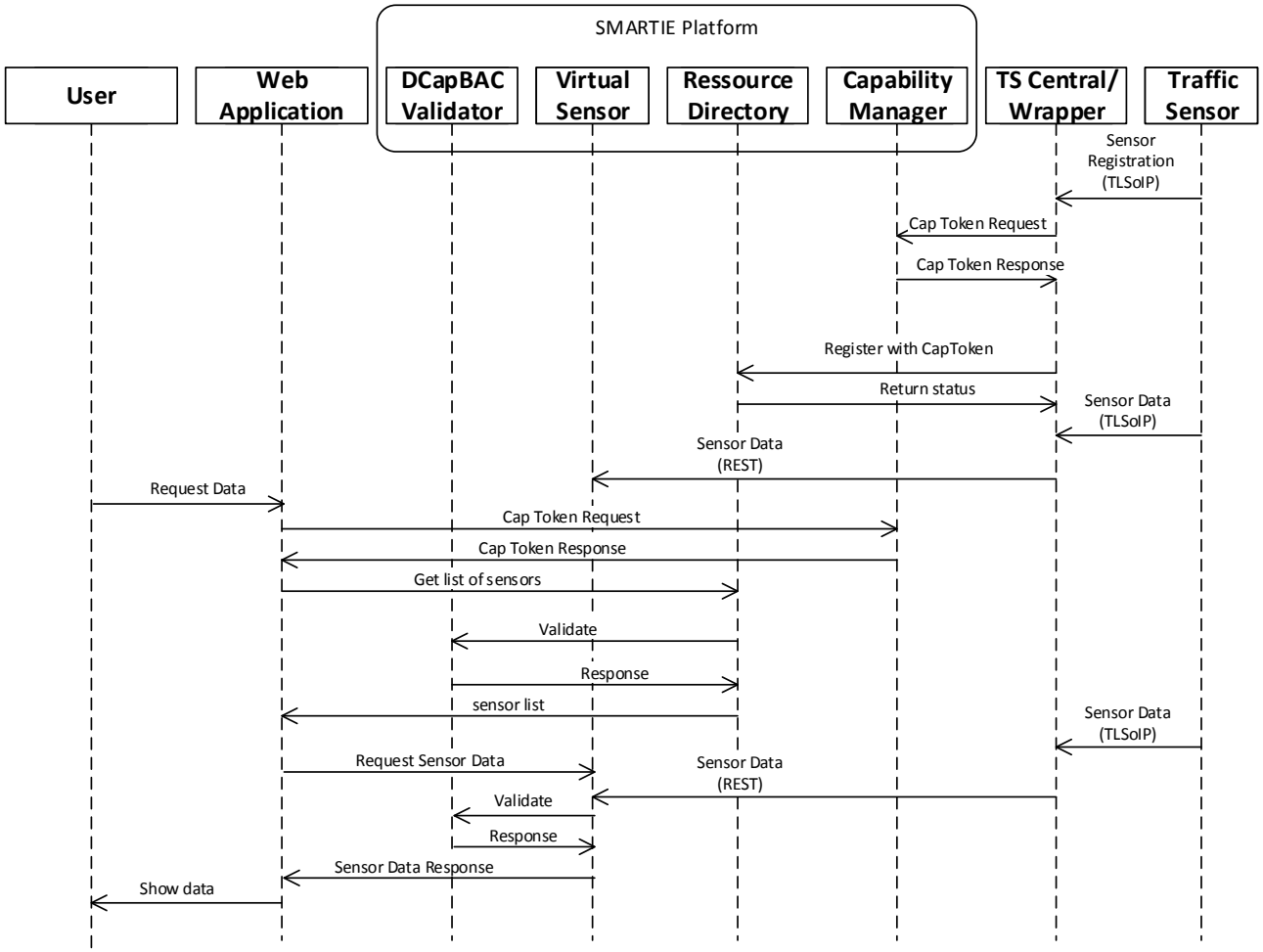

Figure 10: Data Upload and Retrieval for Smart Traffic Control

4.1.9 Validation Testbed for IDS Training

This subsection defines a testbed for the IDS training on a device. In this testbed, the training mode of the IDS is validated. The training period results in a knowledge base / history build up from network traffic as well as detection parameters that are created during this training period. The Knowledge base / history and the according detection parameters describe the monitored network traffic mathematically. The Knowledge base / history and parameters are then used in the IDS detection mode to find unusual behaviour or anomalies that may indicate an attack. Figure 11 shows the interactions between the IDS components.

1. The IDS on the device is started in training mode. The IDS loads initial training parameters and starts the training period.
2. During the training period, the IDS monitors network traffic and generates a knowledge base / history from it. Therefore, the IDS periodically adjusts the parameters to create a mathematical model that describes the monitored traffic.
3. When the training period is finished the IDS stops the adjustment of the knowledge base / history and the parameters.
4. The automatically created detection parameters are inspected for validity and if they describe the monitored traffic within defined limits, i.e. with acceptable uncertainty.
5. The knowledge base / history and the parameters are saved for the detection mode.

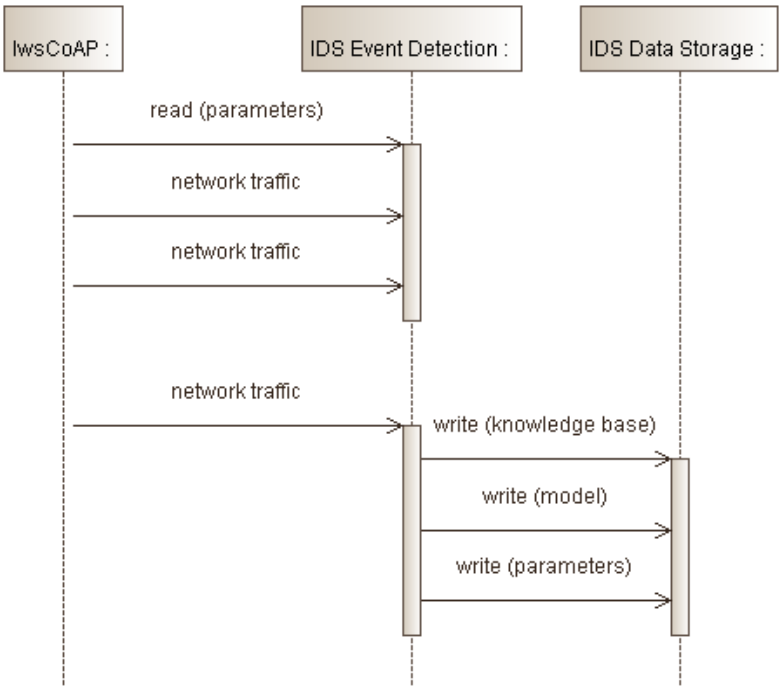

Figure 11: IDS training

4.1.10 Validation Testbed for the IDS

This subsection defines a testbed for the IDS on a device. In this testbed, the detection capability of the IDS is validated. Therefore, a knowledge base / history and according detection parameters are used that were created during a training period. The Knowledge base / history and the detection parameters describe the valid / wanted network traffic mathematically. The Knowledge base / history and the parameters are used by the IDS to compare monitored traffic to the internal mathematical model. That way, the IDS is able to find unusual behaviour or anomalies that may indicate an attack. Traffic that does not match the internal model is

reported as a possible security issue. Intentionally modified network traffic should be detected by the IDS. Figure 12 shows the interactions between the IDS components.

1. The IDS on the device is started in detection mode. The IDS loads the detection parameters and starts the detection.
2. The IDS monitors the network traffic and scans it for unusual behaviour. Therefore, the IDS uses the given knowledge base/history and the detection parameters to compare the monitored network traffic to the internal mathematical model.
3. When unusual network behaviour is detected the IDS reports the issue by creating a detection event with a description of the detected issue.
4. For the validation of the detection capability, intentionally modified network traffic is created and applied to the device to force the IDS to detect and report the issues.
5. The generated events are inspected for validity and if they correctly classify the created malicious traffic.

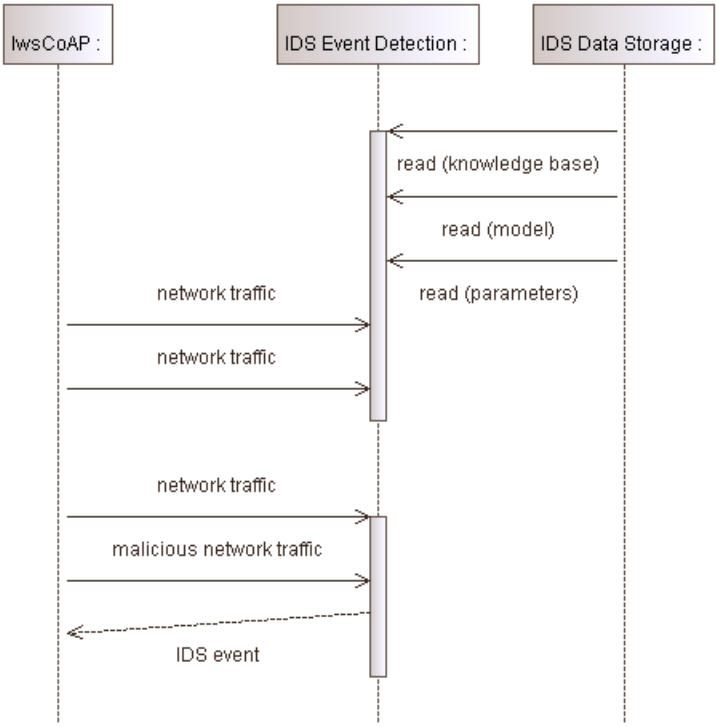

Figure 12: IDS detection mode

4.1.11 Validation Testbed for flow optimisation

Beyond the three scenarios that will be implemented in a real world setting in WP6, SMARTIE is also targeting larger scale IoT scenarios, especially for Smart Cities. For such scenarios, aspects like distributed processing are highly relevant and processing flow optimization is used for optimizing the deployment of the processing components taking into account security constraints that limit the choices where such processing can take place without violating security requirements.

Since SMARTIE partners do not have a large scale Smart City deployment, such a scenario cannot be completely implemented in the real world, but such a deployment can be simulated, allowing the validation of processing flow optimization.

The following components are required for validating the processing flow optimization:

- The Resource Directory (RD) – provides information about information sources and processing nodes
- *The Network Information Directory* – provides information about the network connections and their characteristics
- The Processing Flow Optimization component – creates the optimal mapping of the task plan to processing nodes under security constraints
- The Capability Manager – provides Token for accessing the Processing Flow Optimization component
- *The Processing Unit Deployment component* – deploys processing components according to the mapping created by the Processing Flow Optimization and initializes the processing flows.

As the Processing Flow Component is in the focus of the validation, the Resource Directory and Network Information Directory are pre-configured with artificially created node and network deployments, i.e. the deployments do not exist in the real world and thus the processing flow cannot be actually deployed. The validation will look at performance and scalability of the processing flow optimization and analyse the security aspects.

Figure 13 shows the interactions between the components.

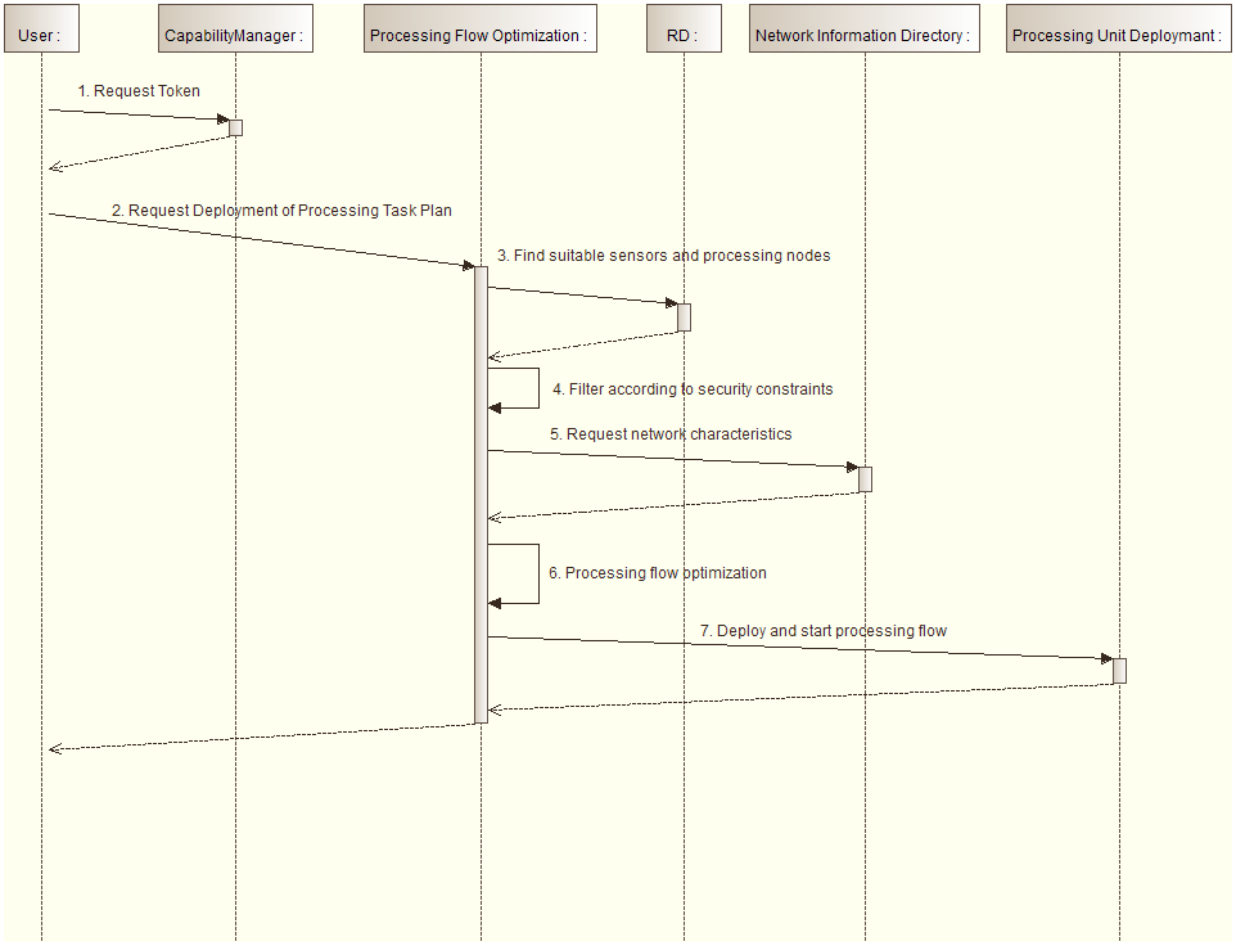

Figure 13: Processing Flow Optimization

1. The user application requests a token from the Capability Manager to access the Processing Flow Optimization component, enabling the Processing Flow Optimization component to discover resources in the Resource Directory and Network Information Directory on behalf of the user.

2. The user application requests the Processing Flow Optimization component to map an abstract processing task plan to concrete sensing and processing resources to be deployed on nodes in the network taking into account security constraints. As part of the request the token is provided.
3. The Processing Flow Optimization component sends discovery requests to the Resource Directory (RD) to find suitable sensors and processing nodes on behalf of the user. RD needs to check authorization based on provided token.
4. The Processing Flow Optimization component filters out nodes that do not fulfil the security constraints
5. *Optional: The Processing Flow Optimization component requests information concerning the interconnections and their delay characteristics for the remaining processing nodes from the Network Information Directory (needed if optimization function takes theses into account) on behalf of the user.*
6. The Processing Flow Optimization finds a suitable mapping of the abstract processing task plan to the sensing and processing resources identified in the previous steps based on an optimization function. Different optimization functions will be considered. As the underlying decision problem of subgraph matching is NP-complete, a heuristic approach may be necessary. The result is an instantiated processing task plan that includes the nodes on which the processing is to be deployed.
7. *The Processing Flow Optimization passes the instantiated processing task plan to the Processing Unit Deployment component, which would deploy the processing components and start the processing flow. However, as this component is not the focus of the validation and there is no actual large scale deployment available, this part will not be evaluated, but could be shown for a demo setting.*

4.2 Integrations involving the SmartData platform

In this section, we will describe the generic flows involving Smartdata platform basic functionalities, as well as the interactions flows involving the Smartdata Platform, upon integration completion with other Smartie components.

4.2.1 Topic subscription

A user can subscribe a topic in order to receive all the data published to it. The Smartdata platform will push all the received data to the topic subscribers.

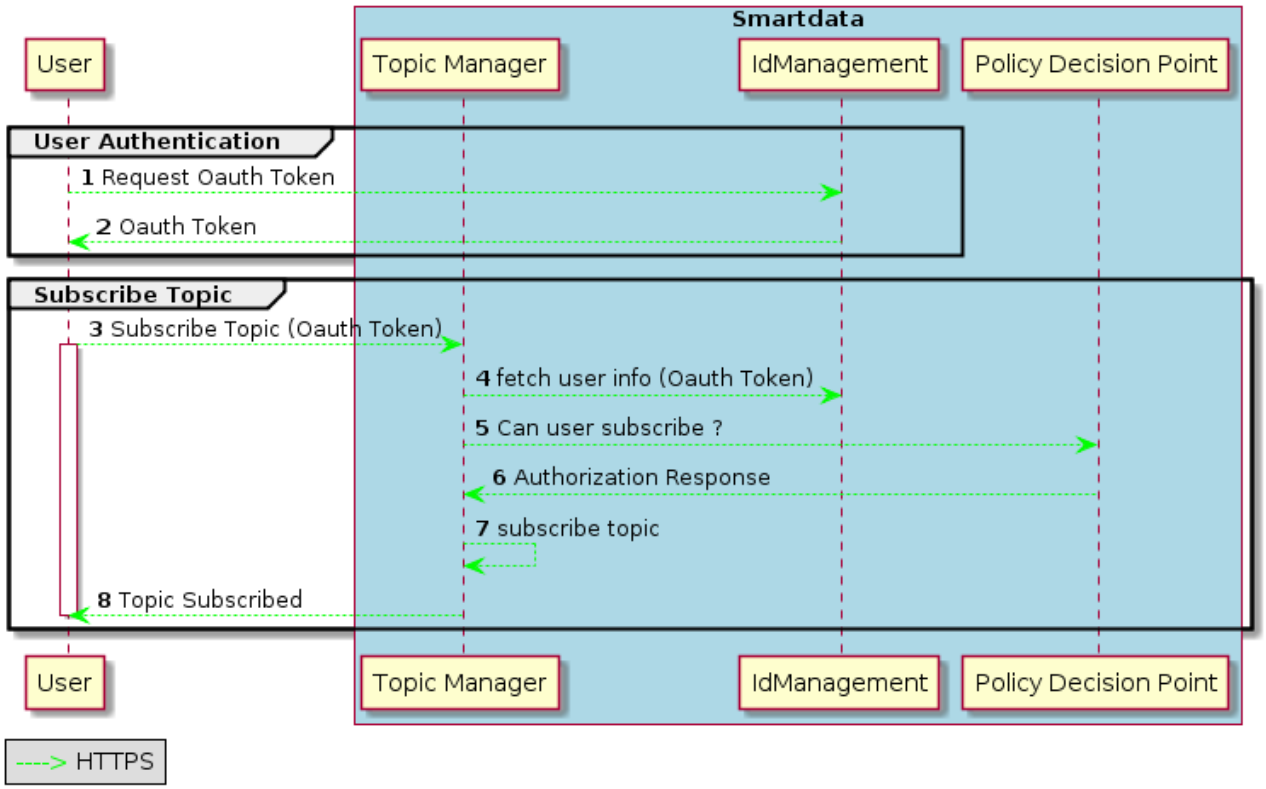

Figure 14: Topic subscription flow

1. The **User** requests and Oauth token from the **IdManagement**;
2. After successfully authenticated, the **IdManagement** will create and send the Oauth token;
3. The **User** subscribes the topic of his choice, using the Oauth token;
4. The **Topic Manger** fetches user info so it can verify if that user is authorized to do such operation;
5. The **Topic Manager** verifies if the **User** can subscribe that topic with the **PDP**;
6. The **PDP** verifies if the **User** is authorized and sends the response to the **Topic Manager**;
7. The topic is subscribed;
8. The **User** is informed that the topic is subscribed.

4.2.2 Requesting Device Data from Topic History

Although a user may subscribe a topic, he can also fetch messages stored in the Smartdata platform.

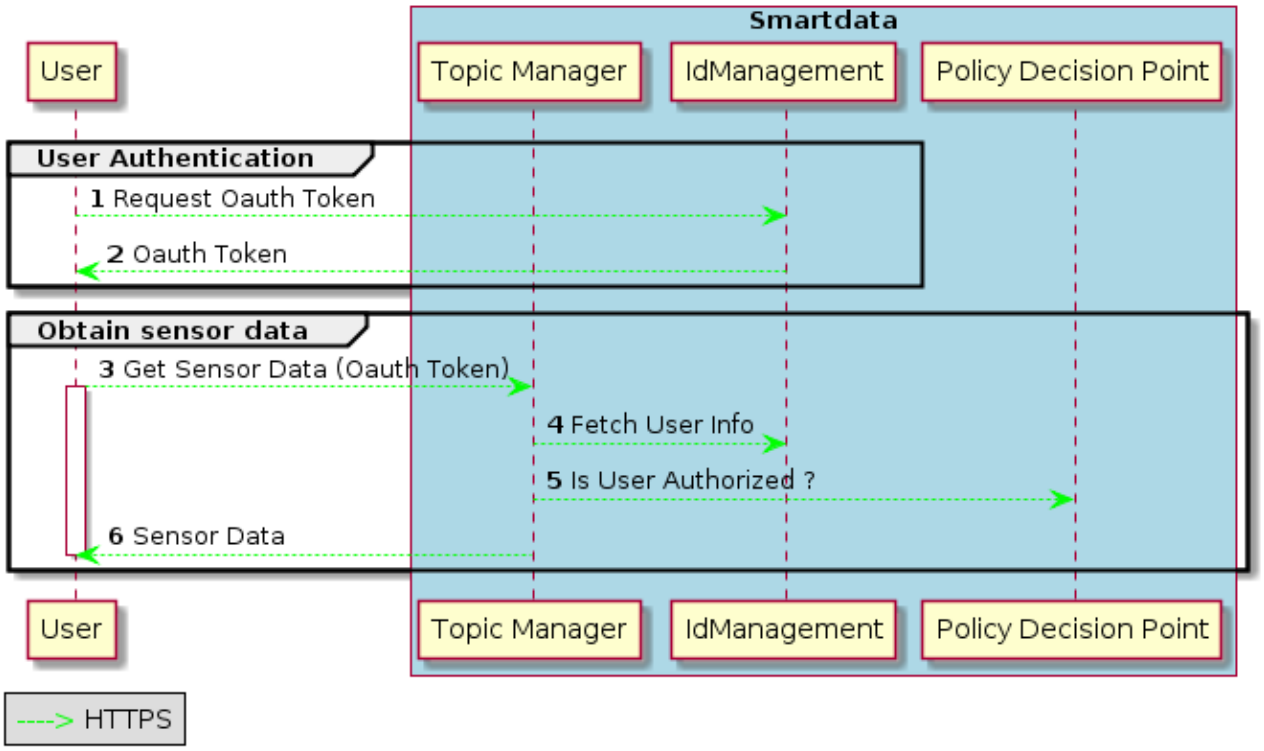

Figure 15: Request Device Data flow

1. The **User** requests and Oauth token from the **IdManagement**;
2. After successfully authenticated, the **IdManagement** will create and send the Oauth token;
3. The **User** requests data from the **Topic Manager** using the Oauth token;
4. The **Topic Manager** will fetch the **User** information from the **IdManagement**.
5. The **Topic Manager** verifies if the **User** is allowed to request data from the platform;
6. The **Topic Manager** sends the data to the **User**.

4.2.3 Device Registration

The device registration is made through the Resource Directory. After registering the device, the Resource Directory creates a topic in the Topic Manager. This will be the topic used by the device to publish data.

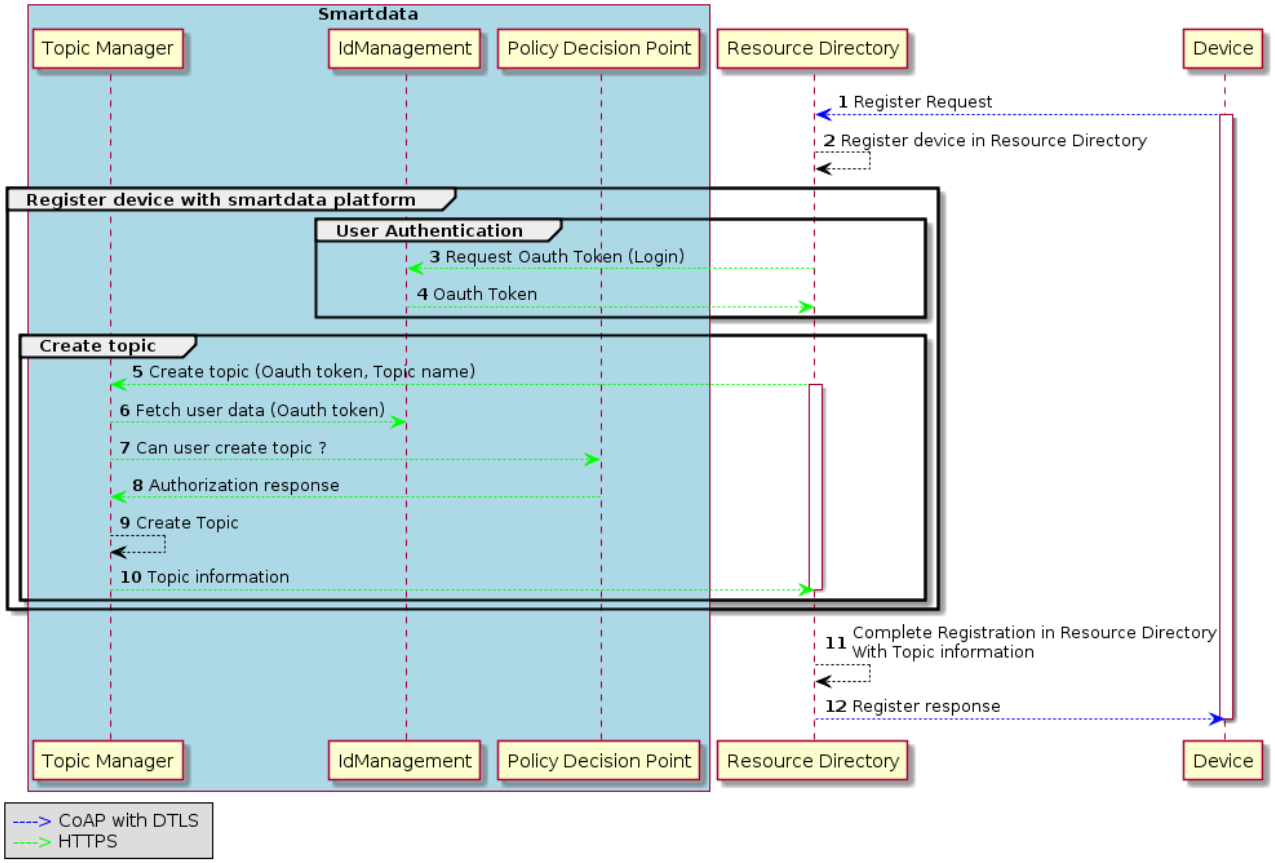

Figure 16: Device Registration flow

1. The **Device** tries to register with the **Resource Directory**;
2. The **Resource Directory** registers the device;
3. The **Resource Directory** requests and Oauth token from the **IdManagement**;
4. After successfully authenticated, the **IdManagement** will create and send the Oauth token;
5. The **Resource Directory** asks the **Topic Manager** to create the topic to where the **Device** will publish data;
6. The **Topic Manager** fetches information about the **Resource Directory** from the **IdManagemet**;
7. The **Topic manager** verifies if the **Resource Directory** can create the topic with the **PDP**;
8. The **PDP** evaluates the request and respond with an authorization response;
9. The **Topic Manager** creates the topic;
10. The **Topic Manager** sends the topic information to the **Resource Directory** in order for it to finish the registration process;
11. The **Resource Directory** completes the registration;
12. The **Resource Directory** sends a register response to the **Device**;

4.2.4 Device Data Publish

This flow describes how a device should proceed in order to publish data to the Topic Manager.

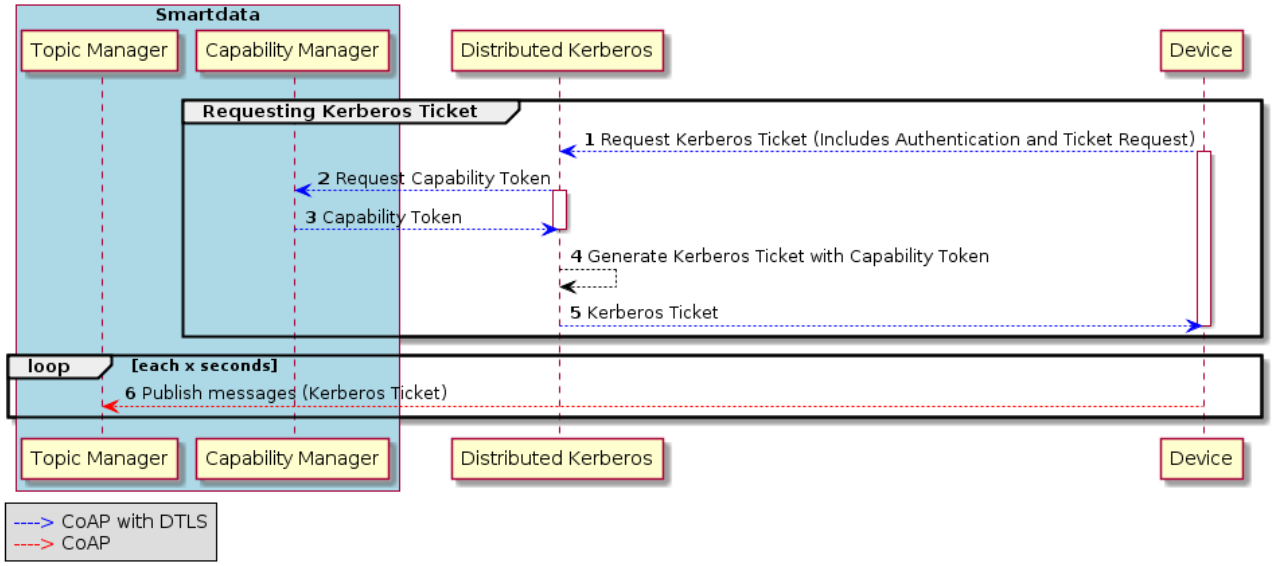

Figure 17: Device Data Publishing flow

1. The **Device** requests a Kerberos ticket from the **Distributed Kerberos**;
2. The **Distributed Kerberos** requests a capability token from the **Capability Manager**;
3. The **Capability Manager** verifies the request and generates a capability token;
4. The **Distributed Kerberos** generates a Ticket;
5. The **Distributed Kerberos** sends the Ticket to the **Device**;
6. The **Device** encrypts the data with the session key and sends it to the **Topic Manager**;

4.2.5 Device Data Pulling

This flow describes how the Topic Manager will pull data from a sensor and publish it to a topic.

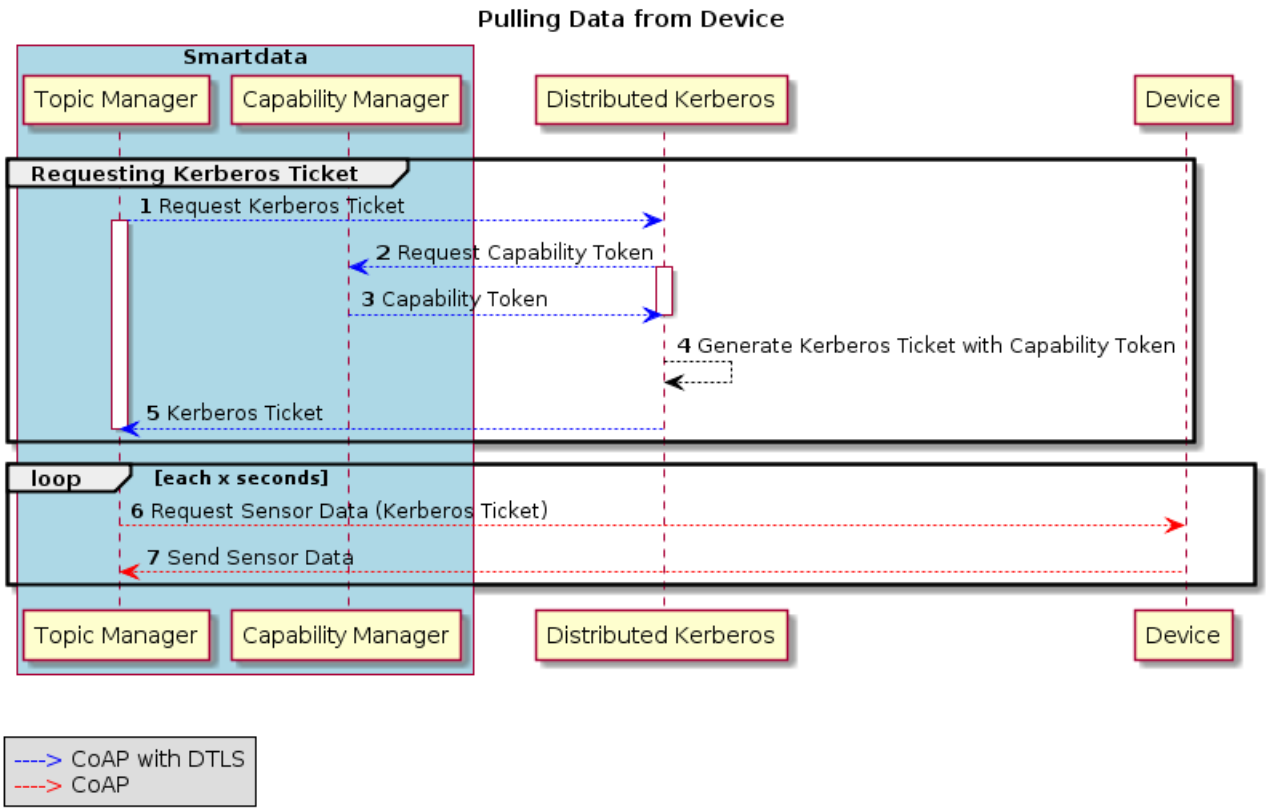

Figure 18: Device Data Pulling flow

1. The **Topic** M**anager** requests a Kerberos ticket from the **Distributed Kerberos**;
2. The **Distributed Kerberos** requests a capability token from the **Capability Manager**;
3. The **Capability Manager** verifies the request and generates a capability token;
4. The **Distributed Kerberos** generates a Ticket;
5. The **Distributed Kerberos** sends the Ticket to the **Topic Manager**
6. The **Topic Manager** encrypts the request with the session key and sends it to the **Device**;
7. The **Device** verifies the request and sends the data to the **Topic Manager**

4.2.6 Device Actuation

The following flow describes how a user can interact with a device using a capability token.

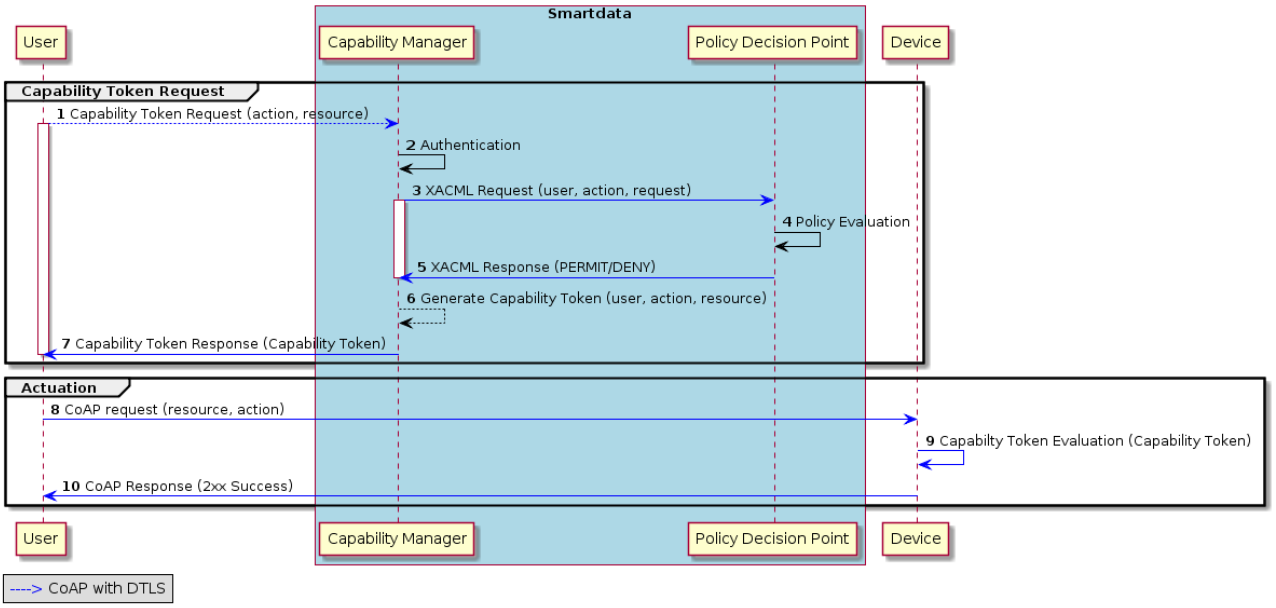

Figure 19: Device Actuation flow

1. The **User** requests a capability token from the **Capability Manager**;
2. The **Capability Manager** authenticates the **User**,
3. The **Capability Manager** verifies with the **PDP** if the **User** can perform such operation;
4. The **PDP** verifies the request against a policy;
5. The **PDP** sends an authorization response to the **Capability Manager**;
6. The **Capability Manager** generates the capability token;
7. The **Capability Manager** sends the token to the **User**;
8. With the capability token the **User** is able to send CoAP request to the **Device**;
9. The **Device** evaluates the token and performs the requested action;
10. The **Device** sends a response to the **User**.

5 Conclusions

This document provides a solid schedule for integrating and testing of innovative SMARTIE components. The validation approaches define will allow for proper assessment of the achieved system properties once the integration is finished. The step wise approach helps to minimize the risk of running in unexpected tricky to handle error situations. We are aware of the fact that realizing the 3 demonstrations in Frankfurt, Murcia and Novi Sad will require integrating existing legacy devices and sensors into the SMARTIE platform. We will identify the existing devices that will be used in each of the demonstrators in D6.1.

6 References

[1] SMARTIE Deliverable D2.1, “Use Cases” http://www.smartie-project.eu/download/D2.1-Use%20Cases.pdf

[2] SMARTIE Deliverable D2.3, “Initial Architecture Specification”

