EUROPEAN COMMISSION

DIGIT
Digital Europe Programme

SMP

Software Architecture Document

Service Metadata Publisher

Version [1.9]

Status [Final]

Date: 28-06-2022

Document Approver(s):

Approver Name	Role
Marco SAMPAIO	Project Manager

Document Reviewers:

Reviewer Name	Role
BERRAH Chaouki	Technical Writer

Summary of Changes:

Version	Date	Created by	Short Description of Changes
V0.1	05/04/2018	Paweł GUTOWSKI	Initial version
V0.5	20/07/2018	Joze RIHTARSIC	Changes after comments from Yves ADAM
V1.0	27/07/2018	Joze RIHTARSIC	Changes after comments from Marcio SAMPAIO
V1.1	30/07/2018	CEF Support	Final version
V1.2	30/09/2018	Caroline AEBY	No more standby service
V1.3	29/10/2018	Joze RIHTARSIC	Updates for 4.1.0.RC release
V1.4	28/11/2018	Joze RIHTARSIC	Updates for 4.1.0.FR release
V1.5	07/10/2019	Joze RIHTARSIC	Updates for 4.1.1. release
V1.6	03/05/2022	Caroline AEBY	No more CEF references + links update
V1.7	19/05/2022	Caroline AEBY	eDelivery support email changed to EC-EDELIVERY
V1.8	02/06/2022	Joze RIHTARSIC Caroline AEBY	Typos and SMP 4.2 RC1
V1.9	28/06/2022	Joze RIHTARSIC Caroline AEBY	SMP 4.2 FR version

Table of Contents

1. Introduction

1.1. Purpose

Service metadata publishing (SMP) was introduced to eDelivery network by PEPPOL project [REF7]. The purpose of the SMP is similar to an address book or business registry. eDelivery participants (message senders and receivers) use SMP to publish their transport/service capabilities and to discover partner's transport/service capabilities as: delivery addresses, supported business processes and document types, etc. The PEPPOLs SMP specification was submitted as input to the OASIS BDXR TC (Business Document Exchange Technical Committee) with the intent of defining a standardized and federated document transport infrastructure for business document exchange. It resulted into a new specification: OASIS Service Metadata Publishing Specification (OASIS SMP specification) [REF1].

The eDelivery Service Metadata Publisher Profile (eDelivery SMP profile) [REF2] provides a set of implementation guidelines for the OASIS SMP specification [REF1]. It is designed to be used in eDelivery with the dynamic receiver (and sender) discovery functionality.

The eDelivery Service Metadata Publisher application (eDelivery SMP) is the sample implementation of the eDelivery SMP profile (thus OASIS SMP spec as well).

This document is the Software Architecture Document of the eDelivery SMP application. It is intended to provide detailed information about the project:

- An overview of the solution
- A description of business and administration functions implemented in the eDelivery SMP
- A description of the application architecture and its modules
- An overview of code organization and code quality measurements
- An overview of technical requirements

1.2. References

Ref.	Document	Content outline
[REF1]	OASIS SMP Specification	Defines documents and REST binding of SMP public interface
[REF2]	eDelivery SMP profile	eDelivery profile of [REF1] specification
[REF3]	eDelivery SMP Administration Guide (pdf) See Documentation section of SMP Software	SMP Administration Guide

Ref.	Document	Content outline
[REF4]	Interface Control Document (pdf) See Documentation section of SMP Software	Defines interface of eDelivery SMP – extends OASIS SMP specification
[REF5]	SML Administration Guide (pdf) See Documentation section of SML Software	Provides comprehensive details on eDelivery SML installation, configuration and maintenance.
[REF6]	eDelivery BDMSL (SML)	Application offered by eDelivery in SaaS model. Facilitates write access to the DNS zone needed for dynamic discovery of Participants. Exposes SOAP interface that is consumed by SMP in order to (un)register participant DNS entries.
[REF7]	PEPPOL	The Pan-European Public Procurement On-Line (PEPPOL) project was a pilot project funded jointly by the European Commission and the PEPPOL Consortium members. After successful completion of the project new organization OpenPEPPOL Association was established. The organization is now responsible for the governance and maintenance of the PEPPOL specifications.

1.3. Definitions

Definition	Description
SMP	Service Metadata Publisher - REST service application providing set of CRUD operations for two web resources: ServiceGroup and ServiceMetadata. SMP is eDelivery implementation of [REF1] and [REF4].
ParticipantIdentifier	The ParticipantIdentifier is an entity that uniquely identifies receiver or sender (participants) in eDelivery process. Examples of identifiers are company registration and VAT numbers, DUNS numbers, GLN numbers, email addresses etc.

Definition	Description
ServiceGroup	The ServiceGroup contains list of services associated with a specific Participant Identifier that is handled by a Service Metadata Publisher. ServiceGroup XML representation is defined by XML Schema attached to [REF1].
ServiceMetadata	The ServiceMetadata contains all necessary metadata (endpoint URLs, certificate for encryption, document types, etc) about a specific service that a participant (service requestor) needs to know in order to send a message to that service. ServiceMetadata XML representation is defined by an XML Schema included into [REF1].
SignedServiceMetadata	ServiceMetadata signed by Service Metadata Publisher (SMP).
DocumentIdentifier	represents document types in a service. It also contains scheme type which represents format of the identifier itself. XML representation is defined by an XML Schema included into [REF1] as part of ServiceMetadata.
BDMSL (SML)	Application offered by eDelivery in SaaS model. Facilitates write access to the DNS zone needed for dynamic discovery of Participants. Exposes a WSDL interface that is consumed by SMP in order to (un)register participants' DNS entries.
Domain	Reflects DNS zones handled by the BDML application. For eDelivery SML the domains are: • acc.edelivery.tech.ec.europa.eu: acceptance domain for testing SMP instances and subdomains. • delivery.tech.ec.europa.eu: production domain.
Subdomain	Subdomain defines business domains handled by BDML application in particular DNS zone. Examples of subdomain (business domain) are: peppol, ehealth, generalerds and they are all in part of domain (DNS zone) edelivery.tech.ec.europa.eu domain.
Dynamic Discovery	Dynamic Discovery is process of discovering participants's service metadata.

2. Overview of the solution

The eDelivery Service Metadata Publisher (eDelivery SMP) enables the participants of an eDelivery Messaging Infrastructure network to dynamically discover each other's capabilities (Legal, Organisational, and Technical). For this to happen, each participant must publish into an SMP its capabilities and settings (including but not limited to):

- business processes that the participant supports
- the security setup (public key certificate)
- the transport protocol (AS2 or AS4)
- the location of the receiver's access point

The SMP usually serves multiple participants to publish theirs exchange capabilities. But in eDelivery network/business domain can coexist in multiple SMPs. Because of this distributed architecture, each participant must have a unique ID in a particular subdomain. A central component, called Business Document Metadata Service Location (BDMSL) [REF6], uses these IDs to create URLs that, when resolved, direct the eDelivery Access Points towards the specific SMP of the participant.

The SMP software component described in this document implements the eDelivery SMP profile [REF2] based on the OASIS Service Metadata Publishing (BDX SMP) [REF1] specifications.

3. Functional View

This section describes interactions, data flows and dependencies between SMP and other integrated applications in dynamic discovery process. All use cases refer to the ICD document (cf. [REF4]), where they are presented with more interface-specific details.

Use cases UC06 – GET ServiceGroup and UC07 – GET ServiceMetadata are implementation of service defined in OASIS SMP Specification [REF1]. All the others use cases cover administration/maintenance services which are not part of the specifications.

The Use cases cover RESTful CRUD operations for following SMP business objects:

ServiceGroup, under relative URL:
/{ParticipantIdentifierScheme}::{ParticipantIdentifier}
ServiceMetadata, under relative URL:
/{ParticipantIdentifierScheme}::{ParticipantIdentifier}/services/{DocTypeIdentifierScheme}::{DocTypeIdentifier}

3.1. Identifiers

3.1.1. Identifiers encoding

According to OASIS SMP Specification [REF1], SMP deals with two types of identifiers: participant and document identifier. The specification [REF1] prescribes that both are built out of scheme and value, delimited by a double-colon separator "::"

ServiceGroup identifier, from business perspective known as Participant Identifier
ServiceGroup identifier := {ParticipantIdentifierScheme}::{ParticipantIdentifier}

ServiceMetadata Identifier, from business perspective known as Document Type Identifier
ServiceMetadata identifier := {DocTypeIdentifierScheme}::{DocTypeIdentifier}

All identifiers that are included in the URL of the REST request must be URL-encoded (note also the double-colon separator "::").

Example: the participant identifier (ServiceGroup identifier) built out of:

- ParticipantIdentifierScheme ="participant#domain#scheme"
- ParticipantIdentifier ="participant#id"

must be encoded in URL request to:
- participant%23domain%23scheme%3A%3Aparticipant%23id

Moreover, in some cases (all PUT requests), the identifiers are present in the URL and in the XML body of the request. In these cases, only identifiers in URL must be URL-encoded.

3.1.2. *ebCore party identifier*

The eDelivery SMP has the feature to support handling participant identifiers as described in eDelivery SMP profile [REF3] in the chapter "Use with eDelivery ebCore Party Identifiers". In this case, the participant starts with the: ***urn:oasis:names:tc:ebcore:partyid-type:*** following by the words: **unregistered** or **iso6523.**

All ebCore party identifiers in the REST request must be URL-encoded using only one double-colon separator ":", as in below example:

- urn:oasis:names:tc:ebcore:partyid-type:iso6523:0088:4035811991021

URL-encoded example:

- urn%23oasis%23names%23tc%23ebcore%23partyid-type%23iso6523%230088%234035811991021

The eDelivery SMP has the option to serialize ebCore party Id to XML according to the OASIS SMP Specification [REF1] as separate values, as in below example:

```
<ParticipantIdentifier scheme="urn:oasis:names:tc:ebcore:partyid-
type:iso6523:0088" >4035811991021</ParticipantIdentifier>
```

or according to the eDelivery SMP profile [REF2] as concatenated value:

```
<ParticipantIdentifier>urn:oasis:names:tc:ebcore:partyid-
type:iso6523:0088:4035811991021</ParticipantIdentifier>
```

The behaviour can be configured and is explained in more details in §5 – "Configuration".

3.1.3. *Identifier's case sensitivity*

SMP can handle identifiers (scheme and value) in case sensitive or in a non-case-sensitive way. The behaviour can be configured: more details can be found in §5 – "Configuration".

When the SMP is configured as non-case-sensitive the SMP normalizes the identifiers extracted from the requests. Identifiers within incoming requests are considered as case insensitive and converted to lowercase. Further processing like the storage and querying in the database is performed using lowercase letters only. If the case-sensitivity configuration is modified, the database records must be updated manually.

When the SMP is configured as case-sensitive then Identifiers are not modified during the whole request processing.

3.2. BDMSL integration

Creation or removal of ServiceGroup within SMP triggers a synchronous (un)registration of relevant record(s) in DNS. This process is required to allow Dynamic Discovery of SMPs to store Participant's metadata.

Write access to DNS zone is facilitated by BDMSL (SML), a centralized application that exposes a SOAP interface for that purpose (cf. [REF6]). SMP is a consumer of the SML services. SML authorization of SMP is based on mutual HTTPS authentication. Therefore, SMP client TLS certificate with private key needs to be configured on SMP side.

If SMP serves data in only one domain, then a single certificate is needed. Otherwise, if the SMP is configured to work in multi-domain mode, the SMP administrator will need to setup one certificate per subdomain. More details can be found in chapter §3.3 – "Domain Multitenancy" and §5 – "Configuration".

3.3. Domain Multitenancy

A SML subdomain can be considered as a set of an inter-network of eDelivery components: SML, SMPs and Access Points for a business domain. All these members communicate with each other within that subdomain and exchange messages according to the strict rules defined for that business domain. One network can be used to exchange invoices between participants, another one could exchange health information between hospitals and insurance companies, etc.

In most scenarios there will be multiple SMPs in a single business domain and each of them will handle ServiceMetadata sets of multiple participants from the same subdomain. The business domain authority can set its own SMP to administrate its participants and the SMP is used only in one domain. But an SMP could be used in more than one business domain at the same time. Because of SML restrictions such setup implies the following SMP functionality:

- The SMP must use a different SMP ID and a different certificate to authenticate for a particular SML subdomain.
- The SMP must be able to sign ServiceMetadata responses using a different certificate for each domain (one certificate per domain).

3.4. Roles

Roles are documented with more details in the ICD document (cf. [REF4]). The table below explains their meaning from a functional perspective:

Role alias	Description
Anonymous	Any user that has not provided any authentication details. This user can query for ServiceGroup and ServiceMetadata.
Admin ServiceGroup	User is assigned at least to one ServiceGroup on which he can perform administrative actions add/update/delete service group extension data and add/update/delete service metadata for the service group.
	User is allowed to execute every administrative action on servicegroup data, extension data and add/update/delete service

Admin SMP	metadata. The user can also define owners and domains for Servicegroup.
System Admin	System user who can administer domains, users, application properties, truststore and keystore on the eDelivery SMP.

3.5. Extensions

To increase security, the eDelivery SMP offers the possibility of registering custom extensions for security scanning/validations of all binary documents such as the certificates and the keystores. The certificates can be uploaded by the users (ServiceGroup and SMP administrators) when setting the user certificate for authentication. The keystores binaries can be uploaded by the SMP administrators when managing the SMP keystore.

When the user loads one of the mentioned payloads, the eDelivery SMP validation framework is activated. At this point, the payload binary data is passed to all registered spring beans, which implement the PayloadValidatorSpi interface below:

```java
package eu.europa.ec.smp.spi;

import eu.europa.ec.smp.spi.exceptions.PayloadValidatorSpiException;

import java.io.InputStream;

/**
 *
 * SMP Service provider interface (SPI) for uploaded payload validation.
 * This SPI interface is intended to allow antivirus validation using third-party
antivirus software.
 */
public interface PayloadValidatorSpi {

    /**
     * Validates the SMP payload. If the payload is invalid the method MUST
     * throw PayloadValidatorSpiException
     *
     * @param payload The payload data to be validated
     * @param mimeType The payload mime type
     * @throws PayloadValidatorSpiException in case the validation does not pass
     */
    void validatePayload(InputStream payload, String mimeType) throws
PayloadValidatorSpiException;
}
```

Listing 1: PayloadValidatorSpi interface

The implementers of the extension must implement the method ***validatePayload*** for payload validation. In the event of malware detection, the method MUST throw the PayloadValidatorSpiException to terminate the future payload handling by the eDelivery SMP.

A simple example of the ***PayloadValidatorSpi*** implementation can be found in the SMP project module ***smp-examples/smp-spi-example/*** (See chapter §4.1).

To register the extension in the eDelivery SMP, the interface implementation class must be

- located under the java package ***eu.europa.ec.smp.spi,***
- tagged with spring bean annotation ***@Component*** or ***@Service***,

as in below example:

```
package eu.europa.ec.smp.spi.example;

import eu.europa.ec.smp.spi.exceptions.PayloadValidatorSpiException;
import org.springframework.stereotype.Service;
import java.io.InputStream;

`@Service
public class ExamplePayloadValidatorSpiImpl implements PayloadValidatorSpi {
    public void validatePayload(InputStream payload, String mimeType) throws
PayloadValidatorSpiException {

          . . .
        }
    }
```

Listing 2: PayloadValidatorSpi implementation example

To prepare the extension for the deployment in the eDelivery SMP, the code must be compiled and stored in the java archive file format known as the JAR.

In the eDelivery SMP, the property ***libraries.folder*** must be configured in the ***smp.conf.properties*** to point to the folder where extension libraries are located. The SMP classloader loads the libraries in the folder at the startup of the SMP and registers the ***PayloadValidatorSpi*** beans.

3.6. UC01 – Manage Administrators

3.6.1. Prerequisites

- User (system admin) has rights to modify content of SMP configuration tables.

3.6.2. Description

This use case does not involve SMP application, instead the user's management is implemented as a simple manual SQL queries. Users and its roles are not cached by the SMP, so they can be used immediately after the corresponding SQL transaction is committed. Sample SQLs inserting users authenticated by password or certificate are presented below. More details on users can be found in §4.3.4 – "Data layer" and §6 - "Security".

```
-- user authenticated with password (oracle dialect)
insert into SMP_USER (ID, USERNAME, PASSWORD, ROLE, ACTIVE, CREATED_ON,
LAST_UPDATED_ON) values (SMP_USER_SEQ.nextval, 'SystemAdmin',
'$2a$10$6nytsush2bqfboliycxn8euvibcnn.wcjurwotjlmndodati85zma ', 'SYSTEM_ADMIN', 1,
sysdate, sysdate);
```

Listing 3 Sample User creation SQL

If the system administrator user is already configured, the system administrator can use the eDelivery SMP UI tool to further manage users.

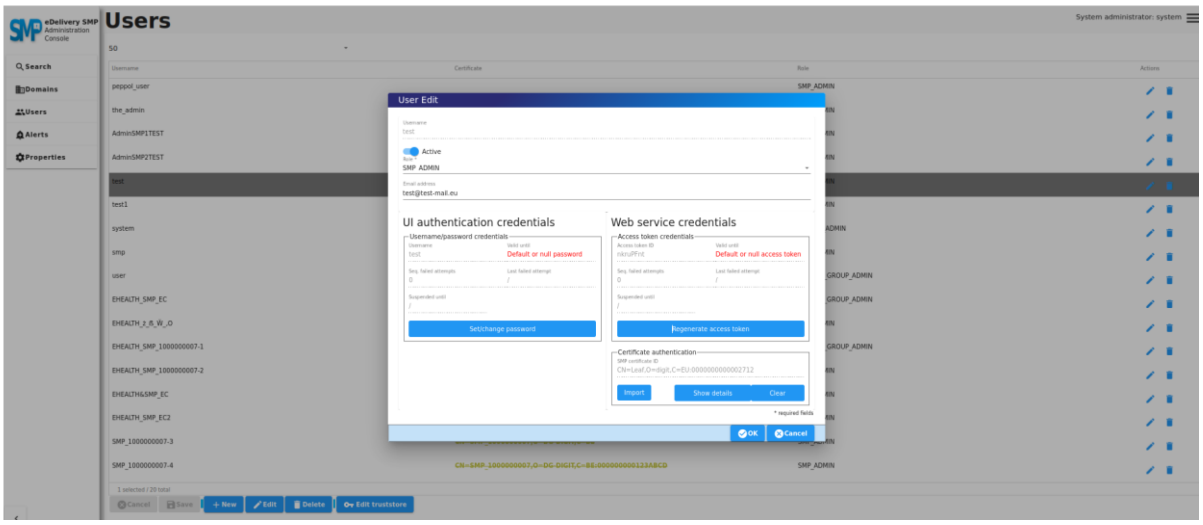


Figure 1: The SMP UI tool for user management

Please note that for Invoking PUT or DELETE Use cases described in sections below, credentials such as Access token or Client certificate must be used for the authentication. Please read chapter §6 - ”Security" for more details.

3.7. UC02 – PUT ServiceGroup (create or update)

3.7.1. Prerequisites

- The authenticated user has the role of "Admin SMP".
- If the ServiceGroup is managed remotely, the "Admin ServiceGroup" must have been created before in the "Administrator" table.
- If the SMP is serving multiple domains, the header field "Domain" must be populated and refer to one of the domains served by the SMP.

3.7.2. Description

"PUT ServiceGroup" is an idempotent[1] create/update REST action. If the SMP is configured to be integrated with BDMSL, then additional synchronous request is performed to register the newly created Participant in the DNS. A sample request is presented below, with the following conventions:

Dark-grey HTTP headers are optional;
Identifiers present in the body of the request and in the URL marked in yellow must match.

Successful responses:

HTTP 200 (OK) – ServiceGroup was updated
HTTP 201 (Created) – New ServiceGroup was created

```
PUT http://smp.eu/participant-domain-scheme%3A%3Aparticipant-id HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
Authorization: Basic c21wX2FkbWluOmNoYW5nZWl0
ServiceGroup-Owner: anotherownerusername
Domain: domain2
Content-Length: 284

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ServiceGroup xmlns="http://docs.oasis-open.org/bdxr/ns/SMP/2016/05">
    <ParticipantIdentifier scheme="participant-domain-scheme">participant-
id</ParticipantIdentifier>
    <ServiceMetadataReferenceCollection/>
</ServiceGroup>
```

Listing 4 Sample PUT ServiceGroup request

[1] as no additional effect if it is called more than once with the same parameters

The SMP administrator can also register a ServiceGroup with the eDelivery UI tool for Service group management (see [1] on the picture below). The ServiceGroup is registered by activating/clicking the save button (see [4] on the picture below) after all the necessary data are entered (see [3] on the picture below).

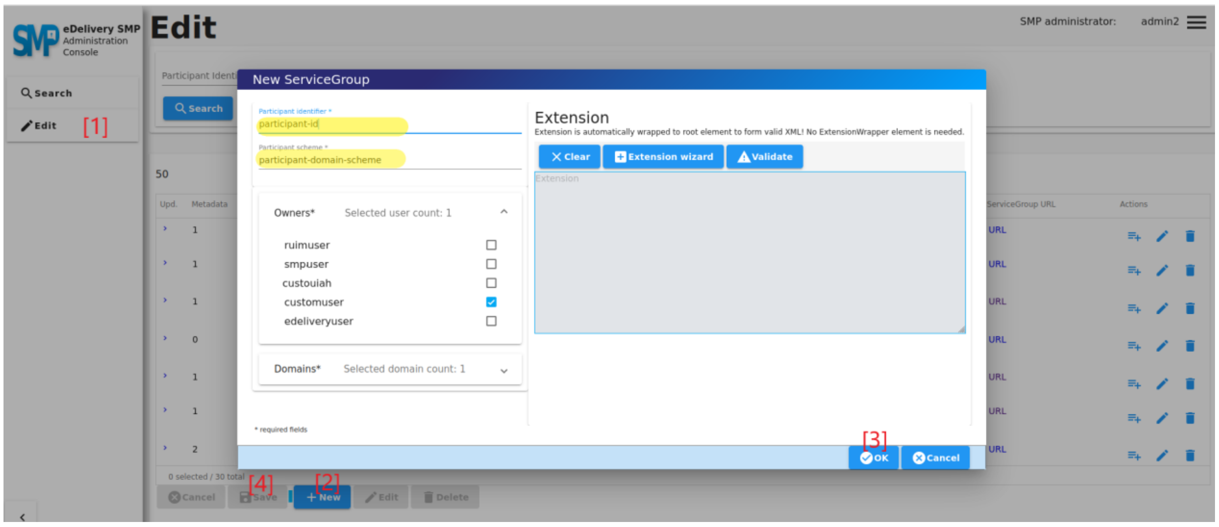
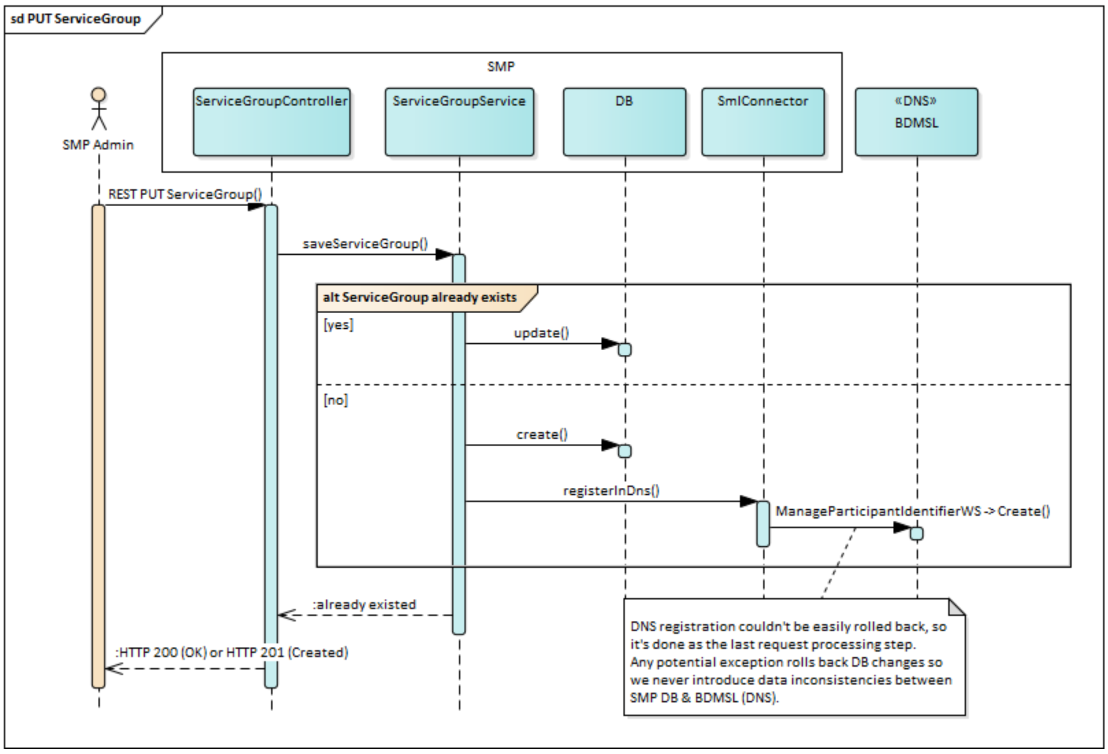


Figure 2: eDelivery SMP UI tool for ServiceGroups management – create/edit

If BDMSL integration is enabled and configured for the selected domain, the SML request is submitted when the ServiceGroup is created.



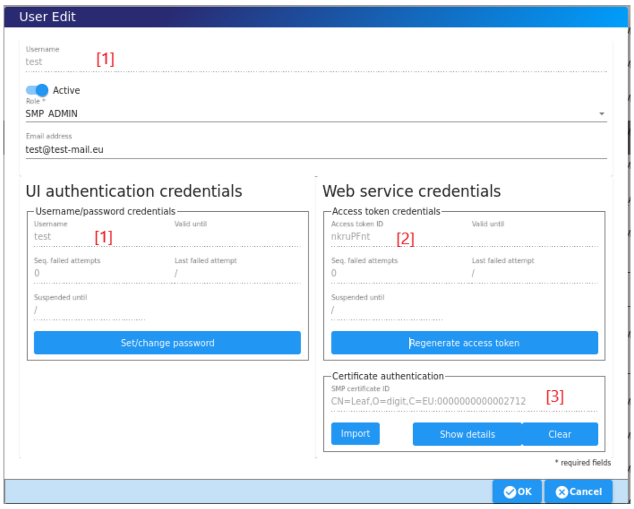
DNS registration couldn't be easily rolled back, so it's done as the last request processing step. Any potential exception rolls back DB changes so we never introduce data inconsistencies between SMP DB & BDMSL (DNS).

Figure 3 PUT ServiceGroup flow

3.7.3. ServiceGroup-Owner HTTP header - Specifying Owner User

Only the SMP administrator has permission to register (or delete) the ServiceGroup. The SMP administrator usually creates a ServiceGroup for the end-user with the "ServiceGroup Administrator" role, which has only the permission to update the ServiceGroup service metadata.

By default, the ServiceGroup is owned by the user who created the ServiceGroup. But this can be changed at creation time by setting the **ServiceGroup-Owner** HTTP header with a different owner's identifier. The identifier of the service owner can be the username (see [1] in the figure below), the access token identifier (see [2] in the figure below), or the certificate identifier (see [3] in the figure below). Below is the image from the eDelivery UI where the data for the user is located:


User Edit

Username: test [1]

Active

Role *: SMP ADMIN

Email address: test@test-mail.eu

UI authentication credentials

Username/password credentials

Username: test [1]	Valid until:
Seq. failed attempts: 0	Last failed attempt: /
Suspended until: /	

Web service credentials

Access token credentials

Access token ID: nkruPFnt [2]	Valid until:
Seq. failed attempts: 0	Last failed attempt: /
Suspended until: /	

Certificate authentication

SMP certificate ID: CN=Leaf,O=digit,C=EU:0000000000002712 [3]

* required fields

Figure 4: Owner identifiers on the User dialog

Below are examples of HTTP header **ServiceGroup-Owner:**

```
ServiceGroup-Owner: anotherownerusername
```

Non-ASCII characters must be URL-encoded, i.e. user **Żółty Jérôme** should be encoded in this way:

```
ServiceGroup-Owner: %C5%BB%C3%B3%C5%82ty%20J%C3%A9r%C3%B4me
```

Users authenticated by certificate can become owners as well, i.e. user **CN=new owner,O=EC,C=BE:000000000000100f** should be encoded:

```
ServiceGroup-Owner: CN%3Dnew%20owner,O%3DEC,C%3DBE%3A000000000000100f
```

3.7.4. Domain HTTP header - Specifying Domain

This feature is used only when the SMP is setup in multi-domain mode. When creating new ServiceGroup the Domain HTTP header must be specified in the PUT ServiceGroup request

```
Domain: domain2
```

More details on Multitenancy can be found in §3.3 – "Domain Multitenancy".

3.8. UC03 - DELETE ServiceGroup

3.8.1. Prerequisites

- The authenticated user has the role of "Admin SMP".
- If the ServiceGroup is managed remotely, the "Admin ServiceGroup" must have been created before in the "Administrator" table.
- If the SMP is serving multiple domains, the header field "Domain" must be populated and refer to one of the domains served by the SMP.

3.8.2. Description

This action removes the specified ServiceGroup from SMP's database **including all related ServiceMetadata**.

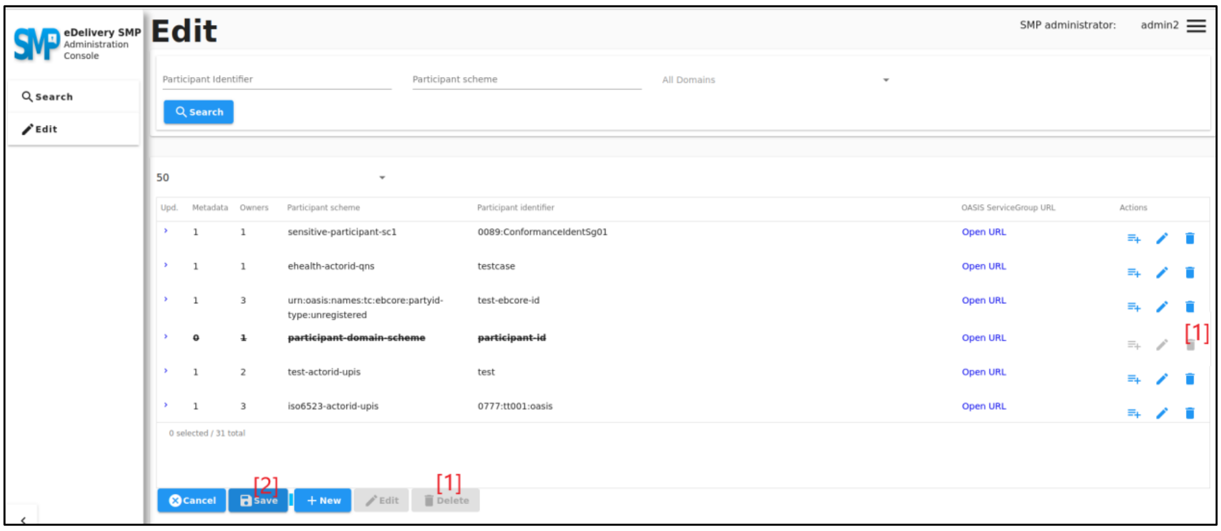
If the SMP is configured to integrate the BDMSL, then an additional synchronous request is issued in order to unregister the Participant from the DNS.

Successful responses:
HTTP 200 (OK) – ServiceGroup was removed

```
DELETE http://smp.eu/participant-domain-scheme%3A%3Aparticipant-id HTTP/1.1
Accept-Encoding: gzip,deflate
Authorization: Basic c21wX2FkbWluOmNoYW5nZWl0
Content-Length: 0
```

Listing 5 Sample delete ServiceGroup request

The SMP administrator can delete a ServiceGroup with the eDelivery UI tool for Service group management. The ServiceGroup can be deleted by selecting the ServiceGroup row, clicking the Delete button (see [1] in the figure below], and then the Save button (see [2] in the figure below).



Upd.	Metadata	Owners	Participant scheme	Participant identifier	OASIS ServiceGroup URL	Actions
›	1	1	sensitive-participant-sc1	0089:ConformanceIdentSg01	Open URL	
›	1	1	ehealth-actorid-qns	testcase	Open URL	
›	1	3	urn:oasis:names:tc:ebcore:partyid-type:unregistered	test-ebcore-id	Open URL	
›	~~0~~	~~1~~	~~participant-domain-scheme~~	~~participant-id~~	Open URL	
›	1	2	test-actorid-upis	test	Open URL	
›	1	3	iso6523-actorid-upis	0777:tt001:oasis	Open URL	

0 selected / 31 total

Figure 5: eDelivery SMP UI tool for ServiceGroups management – delete

If BDMSL integration is enabled and configured for the selected domain, the SML delete request is submitted when the ServiceGroup is deleted.

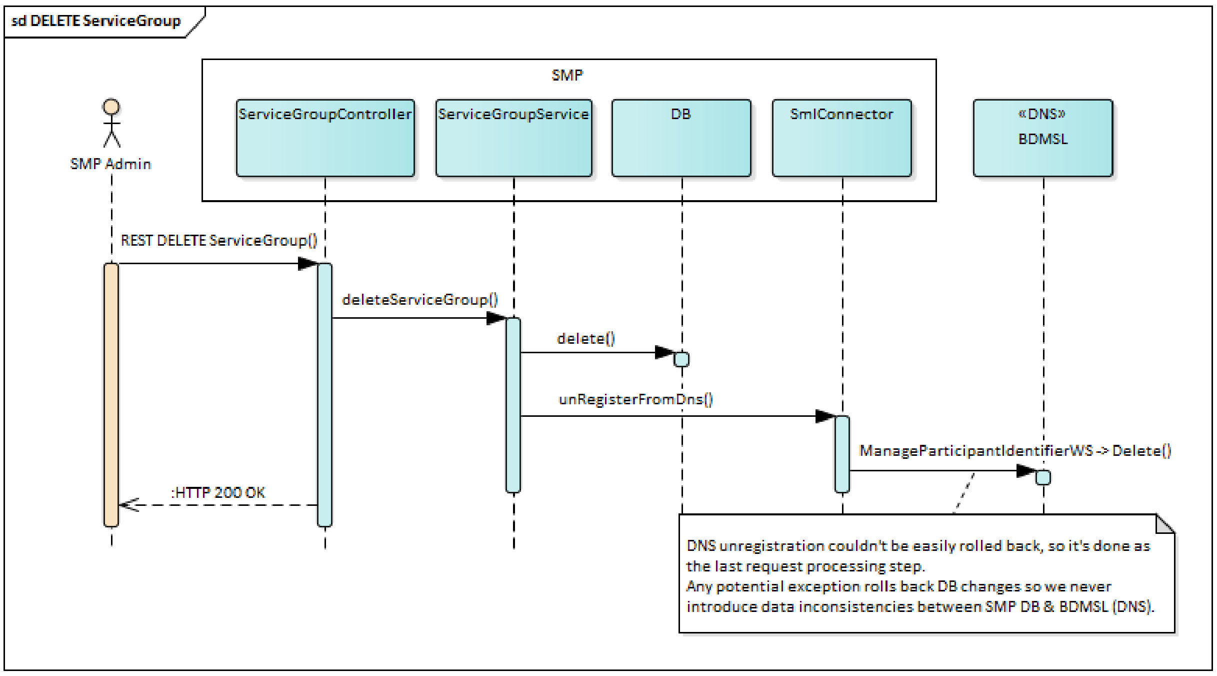

sd DELETE ServiceGroup
SMP
SMP Admin
ServiceGroupController
ServiceGroupService
DB
SmlConnector
«DNS»
BDMSL
REST DELETE ServiceGroup()
deleteServiceGroup()
delete()
unRegisterFromDns()
ManageParticipantIdentifierWS -> Delete()
:HTTP 200 OK
DNS unregistration couldn't be easily rolled back, so it's done as the last request processing step.
Any potential exception rolls back DB changes so we never introduce data inconsistencies between SMP DB & BDMSL (DNS).

Figure 6 DELETE ServiceGroup flow

3.9. UC04 – PUT ServiceMetadata (create or update)

3.9.1. Prerequisites

- The authenticated user has the role of "Admin ServiceGroup" (or "Admin SMP").
- Admin ServiceGroup user initiating the request is linked to the specified ServiceGroup
- The certificate of the "Admin ServiceGroup" is valid.
- The certificate information of the "Admin ServiceGroup" was previously stored in the configuration.

3.9.2. Description

"PUT ServiceMetadata" is an idempotent create/update REST action. A sample request is presented below. Note that Identifiers present in the body of the request and in the URL marked in yellow must match.

ServiceMetadata is processed and stored as the whole unaltered XML document represented as string (including original whitespaces and comments between nodes). ServiceMetadata can be signed by ServiceGroup owner and e-signature can be placed in <Extension> node. To preserve integrity of signed metadata, SMP does not perform any transformation, canonicalization, or decomposing XML document into separate database records. While querying for the metadata (UC07 – GET ServiceMetadata) original XML document is returned.

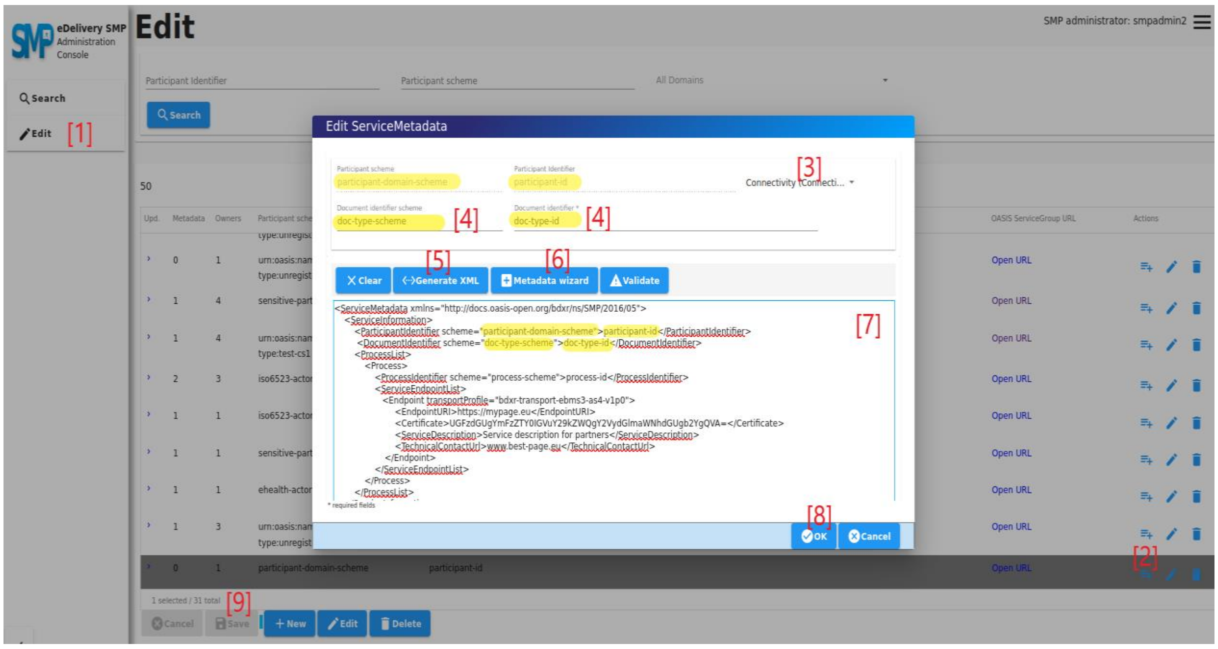
Successful responses:
HTTP 200 (OK) – ServiceMetadata was updated
HTTP 201 (Created) – New ServiceMetadata was created

```
PUT http://smp.eu/participant-domain-scheme%3A%3Aparticipant-id/services/doc-type-scheme%3A%3Adoc-type-id HTTP/1.1

Accept-Encoding: gzip,deflate

Content-Type: text/xml;charset=UTF-8

Authorization: Basic c21wX2FkbWluOmNoYW5nZWl0

Content-Length: 2152

<ServiceMetadata xmlns="http://docs.oasis-open.org/bdxr/ns/SMP/2016/05">

    <ServiceInformation>

        <ParticipantIdentifier scheme="participant-domain-scheme">participant-id</ParticipantIdentifier>

        <DocumentIdentifier scheme="doc-type-scheme">doc-type-id</DocumentIdentifier>

        <ProcessList>

            <Process>

                <ProcessIdentifier scheme=”process-scheme">"process-id</ProcessIdentifier>

                <ServiceEndpointList>

                    <Endpoint transportProfile="busdox-transport-start">

                        <EndpointURI>https://poland.pl/theService</EndpointURI>
                        <RequireBusinessLevelSignature>true
</RequireBusinessLevelSignature>
                        <ServiceActivationDate>2003-01-
01T00:00:00</ServiceActivationDate>
                        <ServiceExpirationDate>2020-05-
01T00:00:00</ServiceExpirationDate>
                        <Certificate>SAMPLEBASE64ENCODEDCERT</Certificate>
                        <ServiceDescription>Sample description of invoicing
service</ServiceDescription>
                        <TechnicalContactUrl>https://example.com
</TechnicalContactUrl>
                    </Endpoint>
                </ServiceEndpointList>
            </Process>
        </ProcessList>
    </ServiceInformation>
</ServiceMetadata>
```

Listing 6 A sample of PUT ServiceMetadata request

The ServiceGroup Owner, with the role ServiceGroup administrator or SMP administrator, can register a ServiceMetadata with the eDelivery UI tool for Service group management (see [1] in picture below). To add ServiceMetadata, click first on “Add ServiceMetadata (see [2] in picture below) for the target ServiceGroup. In the ServiceMetadata dialog, select the Domain (see [3] in picture below) and enter the document scheme and identifier (see [4] in picture below). The XML Document for the service data can be prepared outside the eDelivery SMP and copied and pasted to the text area (see [7] in picture below), or it can be generated using the button “Generate XML” (see [5] in picture below) or using the Metadata Wizard (see [6] in picture below). The prepared XML can be further changed/updated in the text area (see [7] in picture below).

Once the ServiceMetadata XML document is finalized, the ServiceMetadata can be saved to eDelivery SMP by clicking on “OK” (see [8] in picture below) and then “Save” (see [9] in picture below).



Participant scheme: participant-domain-scheme
Participant Identifier: participant-id
Document identifier scheme: doc-type-scheme
Document identifier *: doc-type-id

```
<ServiceMetadata xmlns="http://docs.oasis-open.org/bdxr/ns/SMP/2016/05">
  <ServiceInformation>
    <ParticipantIdentifier scheme="participant-domain-scheme">participant-id</ParticipantIdentifier>
     <DocumentIdentifier scheme="doc-type-scheme">doc-type-id</DocumentIdentifier>
    <ProcessList>
      <Process>
        <ProcessIdentifier scheme="process-scheme">process-id</ProcessIdentifier>
        <ServiceEndpointList>
          <Endpoint transportProfile="bdxr-transport-ebms3-as4-v1p0">
             <EndpointURI>https://mypage.eu</EndpointURI>
             <Certificate>UGFzdGUgYmFzZTY0IGVuY29kZWQgY2VydGlmaWNhdGUgb2YgQVA=</Certificate>
             <ServiceDescription>Service description for partners</ServiceDescription>
             <TechnicalContactUrl>www.best-page.eu</TechnicalContactUrl>
          </Endpoint>
        </ServiceEndpointList>
      </Process>
    </ProcessList>
```

* required fields

Figure 7: eDelivery SMP UI tool for ServiceMetadata management – create/edit

When updating the ServiceMetadata, the old XML document is deleted and replaced with the new XML ServiceMetadata document.

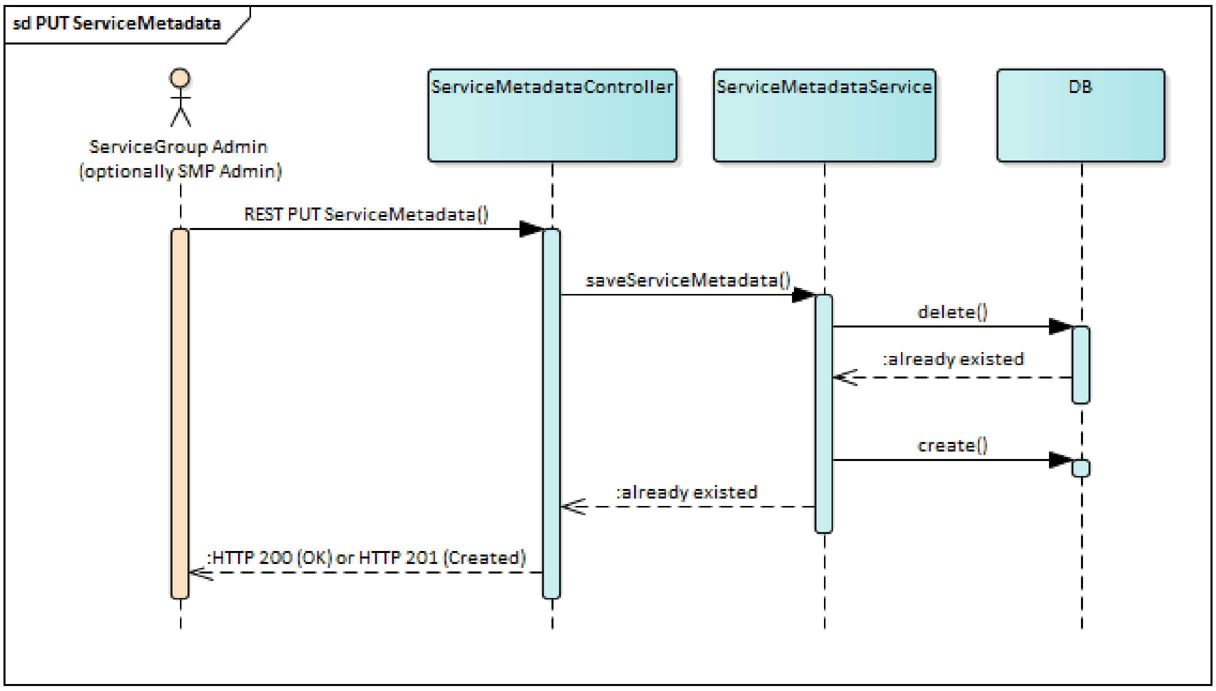

sd PUT ServiceMetadata
ServiceGroup Admin (optionally SMP Admin)
ServiceMetadataController
ServiceMetadataService
DB
REST PUT ServiceMetadata()
saveServiceMetadata()
delete()
:already existed
create()
:already existed
:HTTP 200 (OK) or HTTP 201 (Created)

Figure 8 PUT ServiceMetadata flow

3.10. UC05 – DELETE ServiceMetadata

3.10.1. Prerequisites

- Admin ServiceGroup administrator initiating the request is linked to the specified ServiceGroup (or is "Admin SMP").
- The authenticated user has the role of "Admin ServiceGroup".
- The referenced ServiceMetadata exists.

3.10.2. Description

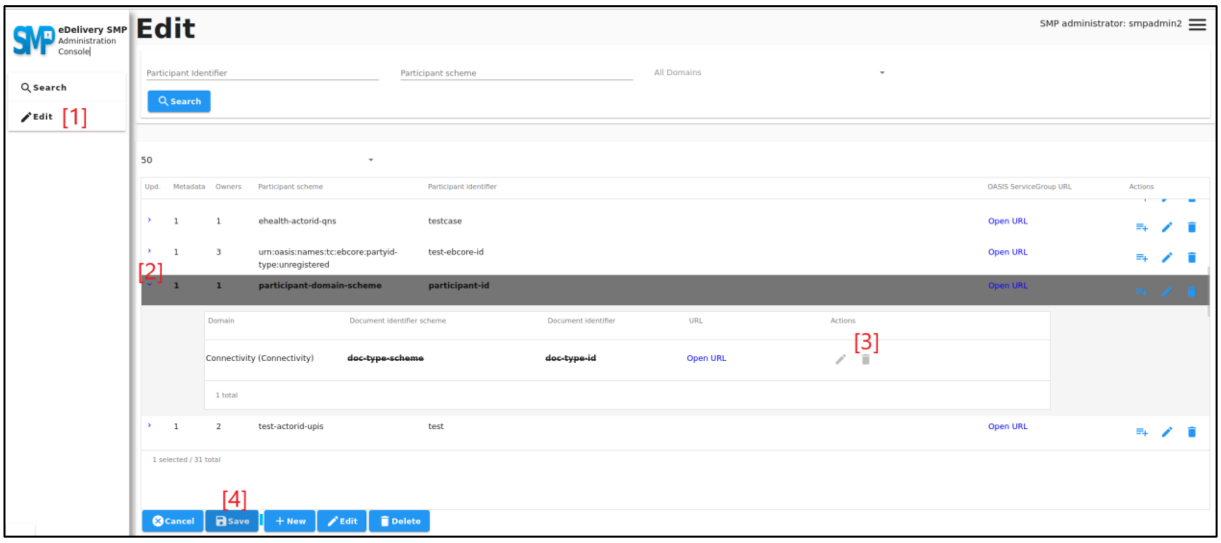
This action removes the specified ServiceMetadata from the SMP's database. The SMP validates the request and deletes corresponding records.

Successful responses:
HTTP 200 (OK) – ServiceGroup was removed

```
DELETE http://smp.eu/participant-domain-scheme%3A%3Aparticipant-id/services/doc-type-scheme%3A%3Adoc-type-id HTTP/1.1

Accept-Encoding: gzip,deflate

Authorization: Basic c21wX2FkbWluOmNoYW5nZWl0

Content-Length: 0
```

Listing 7 Sample DELETE ServiceMetadata request

The ServiceGroup Owner, with the role ServiceGroup administrator or SMP administrator, can delete a ServiceMetadata with the eDelivery UI tool for Service group management (see [1] in picture below). To delete ServiceMetadata, click first to expand the service metadata list for the ServiceGroup (see [2] in picture below), then click on the ServiceMetadata delete button (see [3] in picture below) and then on Save (see [4] in picture below).



Upd.	Metadata	Owners	Participant scheme	Participant identifier	OASIS ServiceGroup URL	Actions
›	1	1	ehealth-actorid-qns	testcase	Open URL	
›	1	3	urn:oasis:names:tc:ebcore:partyid-type:unregistered	test-ebcore-id	Open URL	
˅	1	1	participant-domain-scheme	participant-id	Open URL	
›	1	2	test-actorid-upis	test	Open URL	

Domain	Document identifier scheme	Document identifier	URL	Actions
Connectivity (Connectivity)	~~doc-type-scheme~~	~~doc-type-id~~	Open URL	

1 total

1 selected / 31 total

Figure 9: eDelivery SMP UI tool for ServiceMetadata management – delete

Below is the ServiceMedatada delete flow:

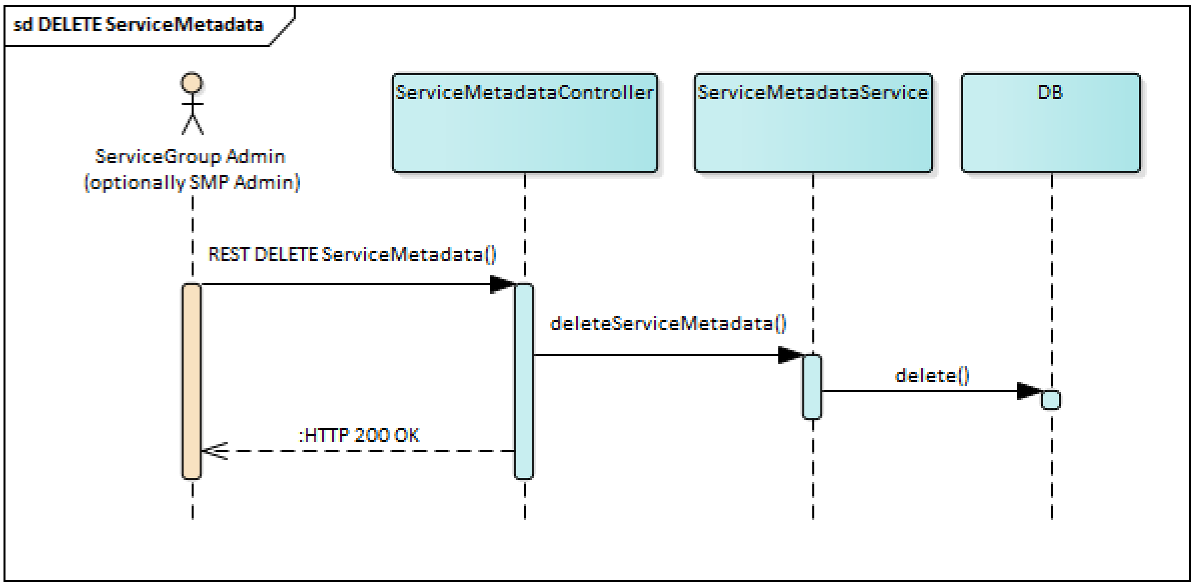

sd DELETE ServiceMetadata
ServiceGroup Admin (optionally SMP Admin)
ServiceMetadataController
ServiceMetadataService
DB
REST DELETE ServiceMetadata()
deleteServiceMetadata()
delete()
:HTTP 200 OK

Figure 10 DELETE ServiceMetadata flow

3.11. UC06 – GET ServiceGroup

3.11.1. Prerequisites

- ServiceGroup exists.

3.11.2. Description

The SMP retrieves the details of the specified ServiceGroup from its database including references to all associated ServiceMetadata and returns them in XML format.

```
GET http://smp.eu/participant-domain-scheme%3A%3Aparticipant-id HTTP/1.1
Accept-Encoding: gzip,deflate
```

Listing 8 Sample GET ServiceGroup request

Successful response:

```
HTTP/1.1 200
Content-Type: text/xml;charset=UTF-8
Content-Length: 496

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceGroup xmlns="http://docs.oasis-open.org/bdxr/ns/SMP/2016/05">
   <ParticipantIdentifier scheme="participant-domain-scheme">participant-
id</ParticipantIdentifier>
   <ServiceMetadataReferenceCollection>
      <ServiceMetadataReference href="http://smp.eu/participant-domain-
scheme%3A%3Aparticipant-id/services/doc-type-scheme%3A%3Adoc-type-id"/>
   </ServiceMetadataReferenceCollection>
</ServiceGroup>
```

Listing 9 Sample GET ServiceGroup response

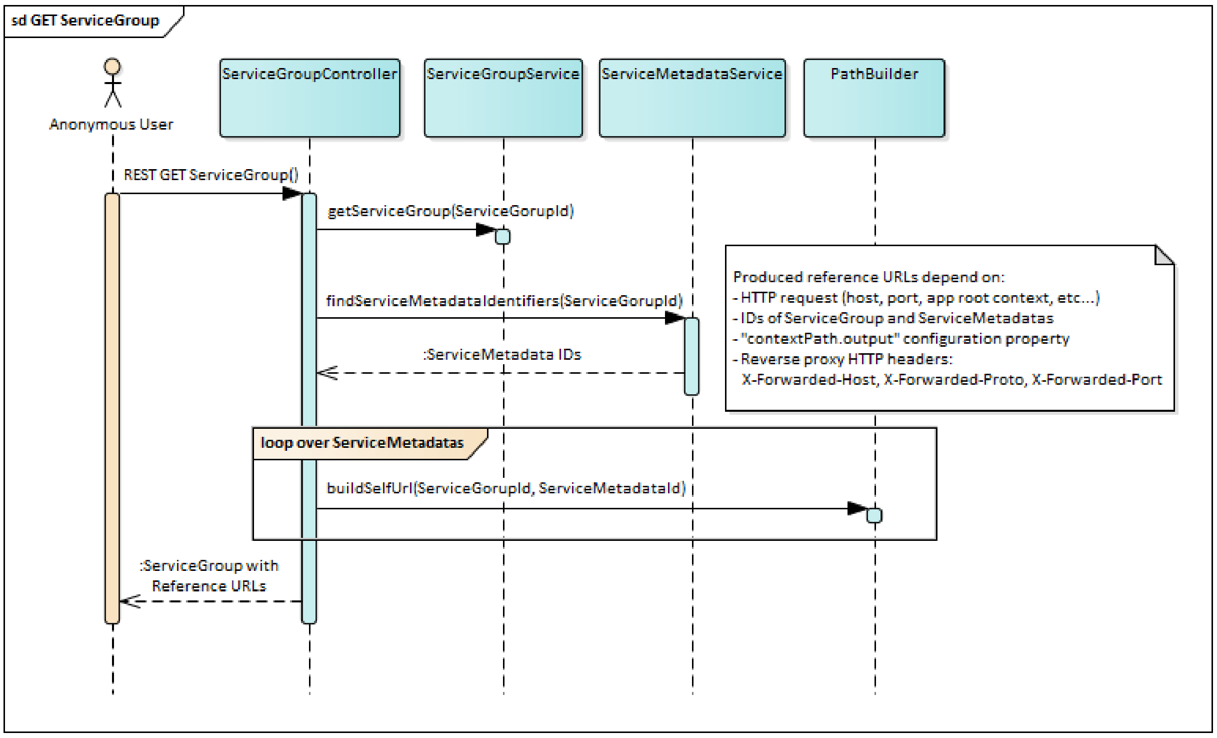

sd GET ServiceGroup
Anonymous User
ServiceGroupController
ServiceGroupService
ServiceMetadataService
PathBuilder
REST GET ServiceGroup()
getServiceGroup(ServiceGorupId)
findServiceMetadataIdentifiers(ServiceGorupId)
:ServiceMetadata IDs
Produced reference URLs depend on:
- HTTP request (host, port, app root context, etc...)
- IDs of ServiceGroup and ServiceMetadatas
- "contextPath.output" configuration property
- Reverse proxy HTTP headers:
X-Forwarded-Host, X-Forwarded-Proto, X-Forwarded-Port
loop over ServiceMetadatas
buildSelfUrl(ServiceGorupId, ServiceMetadataId)
:ServiceGroup with Reference URLs

Figure 11 Get ServiceGroup flow

3.11.3. Reference URLs

The URL references inside the <ServiceMetadataReferenceCollection> node refers to the same SMP and can be immediately used by the client to retrieve ServiceMetadata details. Because the SMP is usually deployed behind a ReverseProxy, when the load balancer or the router redirects the request to the backend system, it adds below listed X-Forwarded-* parameters when constructing the URLs:

- X-Forwarded-Host: identifying the original host requested by the client in the Host HTTP request header, since the host name and/or port of the reverse proxy (load balancer) may differ from the origin server handling the request.
- X-Forwarded-Proto: identifying the originating protocol of an HTTP request, since a reverse proxy (or a load balancer) may communicate with a web server using HTTP even if the request to the reverse proxy is HTTPS.

The ReverseProxy can also hide application root context, for instance, if the application is deployed on the server: http://localhost/smp/. Depending on the ReverseProxy configuration, the application can be accessed from internet without root context: http://smp.eu/ or with root context: http://smp.eu/smp/. To properly build the URL, the parameter contextPath.output must be set accordingly (see chapter §5 –"Configuration").

3.12. UC07 – GET ServiceMetadata

3.12.1. Prerequisites

ServiceMetadata exists in the database.

3.12.2. Description

Service returns details of specified ServiceMetadata from the database. ServiceMetadata is signed and wrapped into the SignedServiceMetadata node.

```
GET http://smp.eu/participant-domain-scheme%3A%3Aparticipant-id/services/doc-
type-scheme%3A%3Adoc-type-id HTTP/1.1
Accept-Encoding: gzip,deflate
```

Listing 10 Sample GET ServiceMetadata request

Successful sample response with SMP's XMLDSIG signature marked in dark-grey:

```
HTTP/1.1 200
Content-Type: text/xml;charset=UTF-8
Content-Length: 4939

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SignedServiceMetadata xmlns="http://docs.oasis-open.org/bdxr/ns/SMP/2016/05">
   <ServiceMetadata>
      <ServiceInformation>
        <ParticipantIdentifier scheme="participant-domain-scheme">participant-
id</ParticipantIdentifier>
        <DocumentIdentifier scheme="doc-type-scheme">doc-type-
id</DocumentIdentifier>
         <ProcessList>
            <Process>
               <ProcessIdentifier scheme="cenbii-procid-
ubl">urn:www.cenbii.eu:profile:bii04:ver1.0</ProcessIdentifier>
               <ServiceEndpointList>
                  <Endpoint transportProfile="busdox-transport-start">
                     <EndpointURI>https://poland.pl/theService</EndpointURI>
                     <RequireBusinessLevelSignature>true
</RequireBusinessLevelSignature>
                     <ServiceActivationDate>2003-01-
01T00:00:00</ServiceActivationDate>
                     <ServiceExpirationDate>2020-05-
01T00:00:00</ServiceExpirationDate>
                     <Certificate>BASE64ENCODEDSAMPLECERT</Certificate>
                     <ServiceDescription>Sample description of invoicing
service</ServiceDescription>
```

```
                     <TechnicalContactUrl>https://example.com
</TechnicalContactUrl>

                  </Endpoint>

               </ServiceEndpointList>

            </Process>

         </ProcessList>

      </ServiceInformation>

   </ServiceMetadata>

   <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

      <SignedInfo>

         <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315"/>

         <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-
sha256"/>

         <Reference URI="">

            <Transforms>

               <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>

            </Transforms>

            <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>

            <DigestValue>BASE64SAMPLEDIGEST</DigestValue>

         </Reference>

      </SignedInfo>

      <SignatureValue>BASE64SAMPLESIGNATUREVALUE</SignatureValue>

      <KeyInfo>

         <X509Data>

            <X509SubjectName>Certificate subject name</X509SubjectName>

            <X509Certificate>BASE64CERTUSEDFORSIGNING</X509Certificate>

         </X509Data>

      </KeyInfo>

   </Signature>

</SignedServiceMetadata>
```

Listing 11 Sample GET ServiceMetadata response

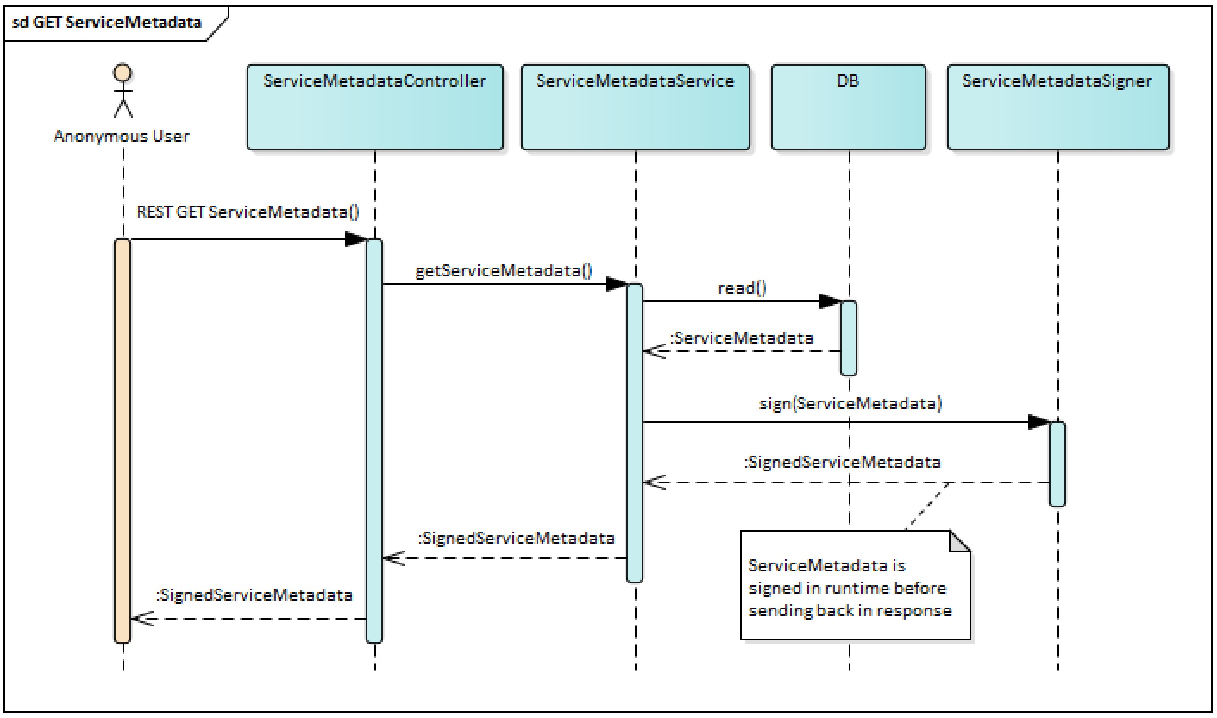


Figure 12 GET ServiceMetadata flow

4. Implementation View

4.1. Source code and modules overview

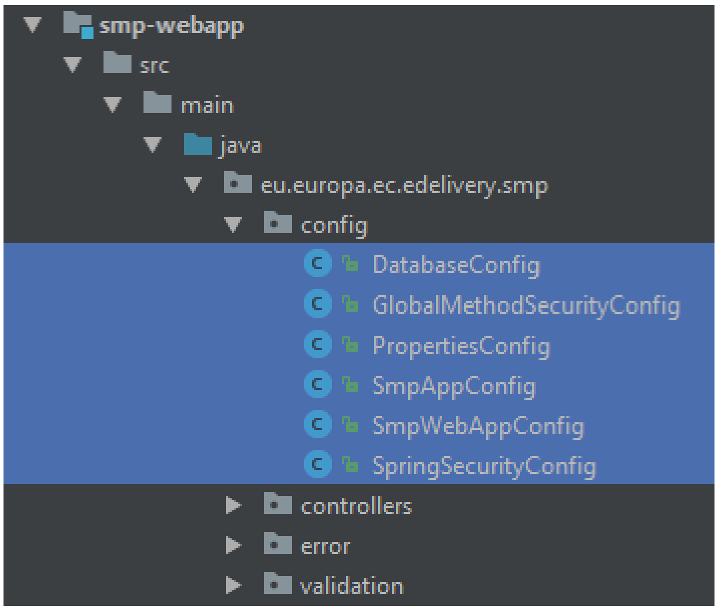
The SMP is a Java REST application packaged in a WAR file. Dependencies and build are organised with Maven 3. Below is description of maven submodules.

Module	Description
smp-api	Module contains OASIS SMP response schemas and administration API schemas. Module purpose is to generate java API classes from predefined XML schemas. Module also contains utility classes used for conversion and validation. This module is used by the SMP REST service implementation and can also be used for building SMP client.
smp-parent-pom	Parent POM contains dependency and plugin management used in sub-modules.
smp-angular	Angular web fragment for UI.
smp-server-library	SMP core library. Covers database access and business logic. This module does not have any HTTP/REST dependencies.
smp-soapui-tests	Module contains Soap UI tests for regression testing in CI server.
smp-ui-tests	Module contains regression tests for ui.
smp-webapp	REST interface over the core library. Defines REST binding, adds web-specific validations and security. Module also build SMP artefact for deploying to application server and package SMP setting examples, it's output is WAR application and ZIP file smp_setup.zip with configuration files and Soap UI test project.
smp-docker	Module contains files for building docker images for weblogic/oracle and mysql/tomcat setup. Project also contains compose files to start the setups. The main purpose of the module is to prepare the environment for API and UI integrational testing.
smp-examples	The module contains SMP examples of API and SPI implementations.

	Currently, SPI payload validation example.

4.2. Application skeleton - Spring annotations context setup

The SMP application is built with SpringFramework, the context is setup by classes with @Configuration annotations which are organized hierarchically. List of configuration classes, sample classes defining dependencies, scanning rules in packages and importing another context configuration are presented below.



```
▼ smp-webapp
    ▼ src
        ▼ main
            ▼ java
                ▼ eu.europa.ec.edelivery.smp
                    ▼ config
                          DatabaseConfig
                          GlobalMethodSecurityConfig
                          PropertiesConfig
                          SmpAppConfig
                          SmpWebAppConfig
                          SpringSecurityConfig
                    ▶ controllers
                    ▶ error
                    ▶ validation
```

Figure 13 List of context configuration classes

```
@Configuration
@ComponentScan(basePackages = {
        "eu.europa.ec.edelivery.smp.validation",
        "eu.europa.ec.edelivery.smp.services",
        "eu.europa.ec.edelivery.smp.sml",
        "eu.europa.ec.edelivery.smp.conversion"})
@Import(DatabaseConfig.class)
public class SmpAppConfig {}
```

Listing 12 Sample context configuration class

4.3. Layers overview

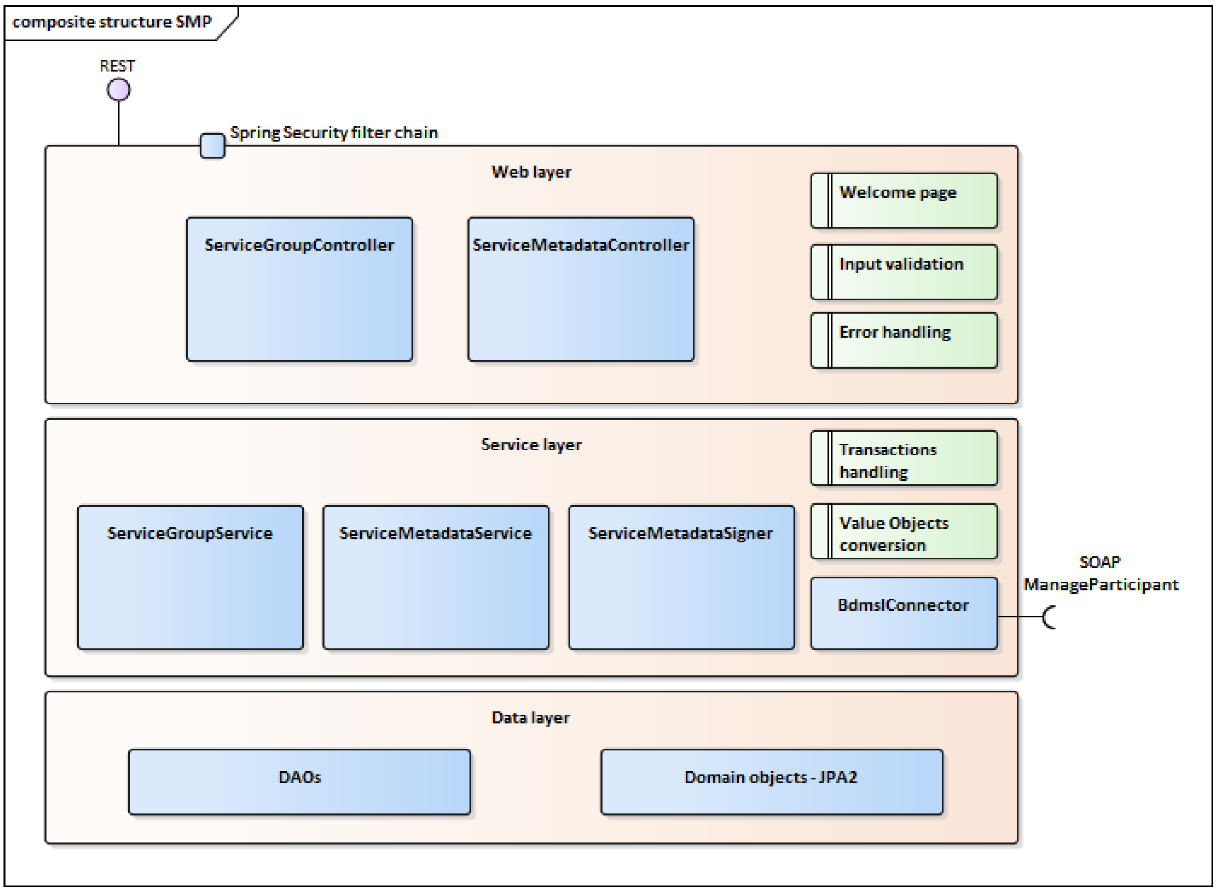


Figure 14 SMP layers structure

4.3.1. Spring MVC - REST interface layer

The top layer, implemented within the smp-webapp module, uses Spring MVC's framework. Both of the resources (ServiceGroup, ServiceMetadata) have a dedicated Controller implementation. Each controller has 3 public methods (GET, PUT, and DELETE) which share the same URL defined by *@RequestMapping* annotation at the Controller class level.

A sample method definition, utilizing also metadata transferred in the request headers is presented below.

This layer is responsible for: REST binding, security validation (more details in §6 – "Security"), request data validation, forwarding request to services layer and forwarding response back to the caller and for error handling.

```
@RestController
@RequestMapping("/{serviceGroupId}")
public class ServiceGroupController {
    @PutMapping
    @Secured("ROLE_SMP_ADMIN")
    public ResponseEntity saveServiceGroup(
            @PathVariable String serviceGroupId,
            @RequestHeader(name = "ServiceGroup-Owner", required = false) String
serviceGroupOwner,
            @RequestHeader(name = "Domain", required = false) String domain,
            @RequestBody String body) throws XmlInvalidAgainstSchemaException,
UnsupportedEncodingException { /* . . . */ }
```

Listing 13 Sample method implementing REST action

4.3.2. Business Services layer

The business logic is implemented within the *smp-server-library module*. Business logic is implemented as ServiceGroup and ServiceMetadata Services. Module contains additional classes for Integration with BDMSL, signing messages and transaction handling with use of Spring *@Transactional* annotation and TransactionManager.

Because the SMP is a small application without need of polymorphism, the implementation does not use interface patterns for its services.

Sample Service method definition is presented below:

```
@Service
public class ServiceMetadataService {
    @Transactional
    public boolean saveServiceMetadata(ParticipantIdentifierType serviceGroupId,
DocumentIdentifier documentId, String xmlContent) { /* . . . */ }
```

Listing 14 Sample transactional Service method

4.3.2.1. BDMSL Integration

The BDMSL integration used by *ServiceGroupService* is implemented by *BDMSLConnector*. Participant's (un)registration is called synchronously as the last action Service's method to make sure that any potential *RuntimeException* causes rollback of the whole transaction, including database changes.

To support multiple domains functionality (See chapter §3.3 – "Domain Multitenancy") *BDMSLClientFactory* was introduced. Its responsibility is to create and preconfigure client (*BDMSLConnector*) to set up needed HTTP headers, configure proxy, manage client X509 Certificate, for each particular domain.

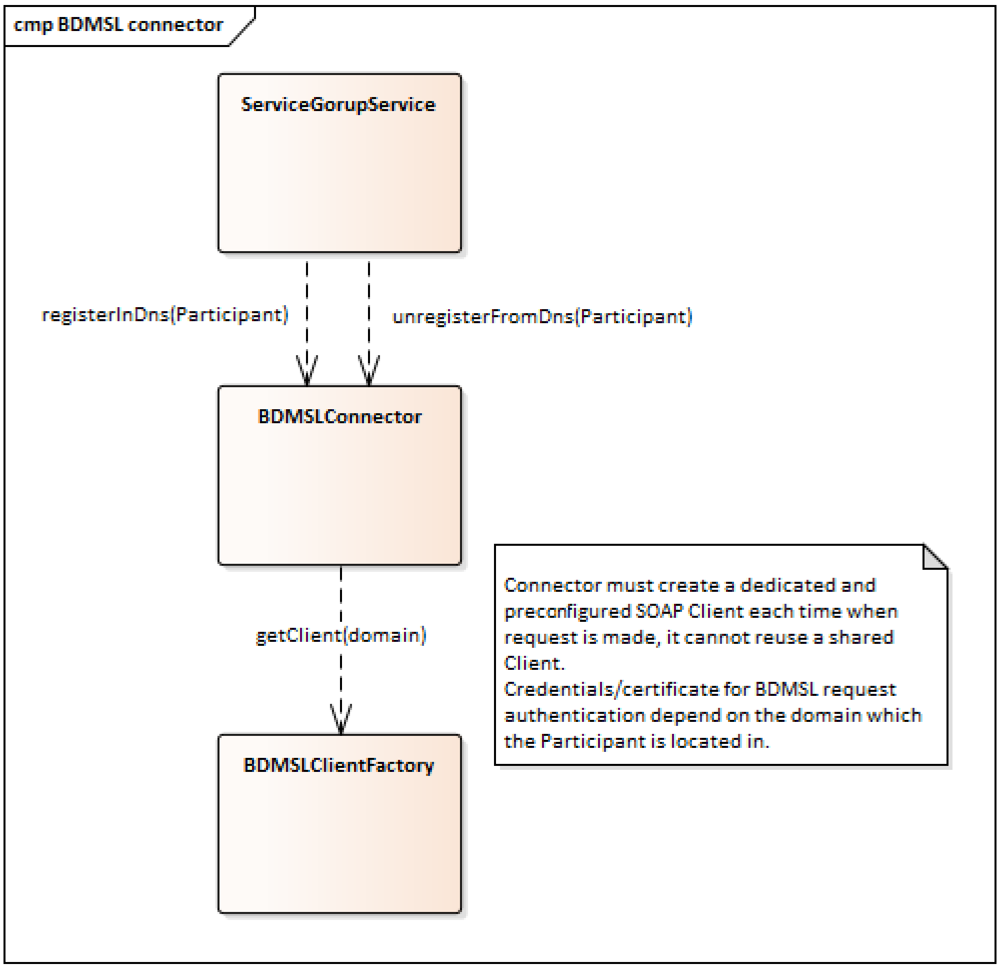

cmp BDMSL connector
ServiceGorupService
registerInDns(Participant)
unregisterFromDns(Participant)
BDMSLConnector
getClient(domain)
BDMSLClientFactory
Connector must create a dedicated and preconfigured SOAP Client each time when request is made, it cannot reuse a shared Client.
Credentials/certificate for BDMSL request authentication depend on the domain which the Participant is located in.

Figure 15 BDMSLConnector needs a dedicated client depending on the Domain used

4.3.2.2. *Case (in)sensitivity normalisation*

4.3.3. *Case (in)sensitivity support, as functionally described in §4.3.2.2 –"ebCore party identifier*

The eDelivery SMP has the feature to support handling participant identifiers as described in eDelivery SMP profile [REF3] in the chapter "Use with eDelivery ebCore Party Identifiers". In this case, the participant starts with the: ***urn:oasis:names:tc:ebcore:partyid-type:*** following by the words: **unregistered** or **iso6523.**

All ebCore party identifiers in the REST request must be URL-encoded using only one double-colon separator ":", as in below example:

- urn:oasis:names:tc:ebcore:partyid-type:iso6523:0088:4035811991021

URL-encoded example:

- urn%23oasis%23names%23tc%23ebcore%23partyid-type%23iso6523%230088%234035811991021

The eDelivery SMP has the option to serialize ebCore party Id to XML according to the OASIS SMP Specification [REF1] as separate values, as in below example:

```
<ParticipantIdentifier scheme=”urn:oasis:names:tc:ebcore:partyid-
type:iso6523:0088” >4035811991021</ParticipantIdentifier>
```

or according to the eDelivery SMP profile [REF2] as concatenated value:

```
<ParticipantIdentifier>urn:oasis:names:tc:ebcore:partyid-
type:iso6523:0088:4035811991021</ParticipantIdentifier>
```

The behaviour can be configured and is explained in more details in §5 – "Configuration".

Identifier's case sensitivity" and §5 – "Configuration" is implemented by the *CaseSensitivityNormalizer* bean. Normalization is performed at the very beginning of each service method processing. Moreover, by separating this to a dedicated bean, normalization can be used as well for permissions verification in connection with Spring Security's *@PreAuthorize anotation*:

```
@PreAuthorize("hasAnyAuthority('ROLE_SMP_ADMIN',
@caseSensitivityNormalizer.normalizeParticipantId(#serviceGroupId))")
```

Listing 15 Sample use of CaseSensitivityNormalizer inside of the @PreAuthorize annotation

4.3.4. Data layer

The SMP stores data in a relational database. MySQL and Oracle DDL scripts are released with the application in *smp-setup.zip* file. The database object relations are presented in the following figure:

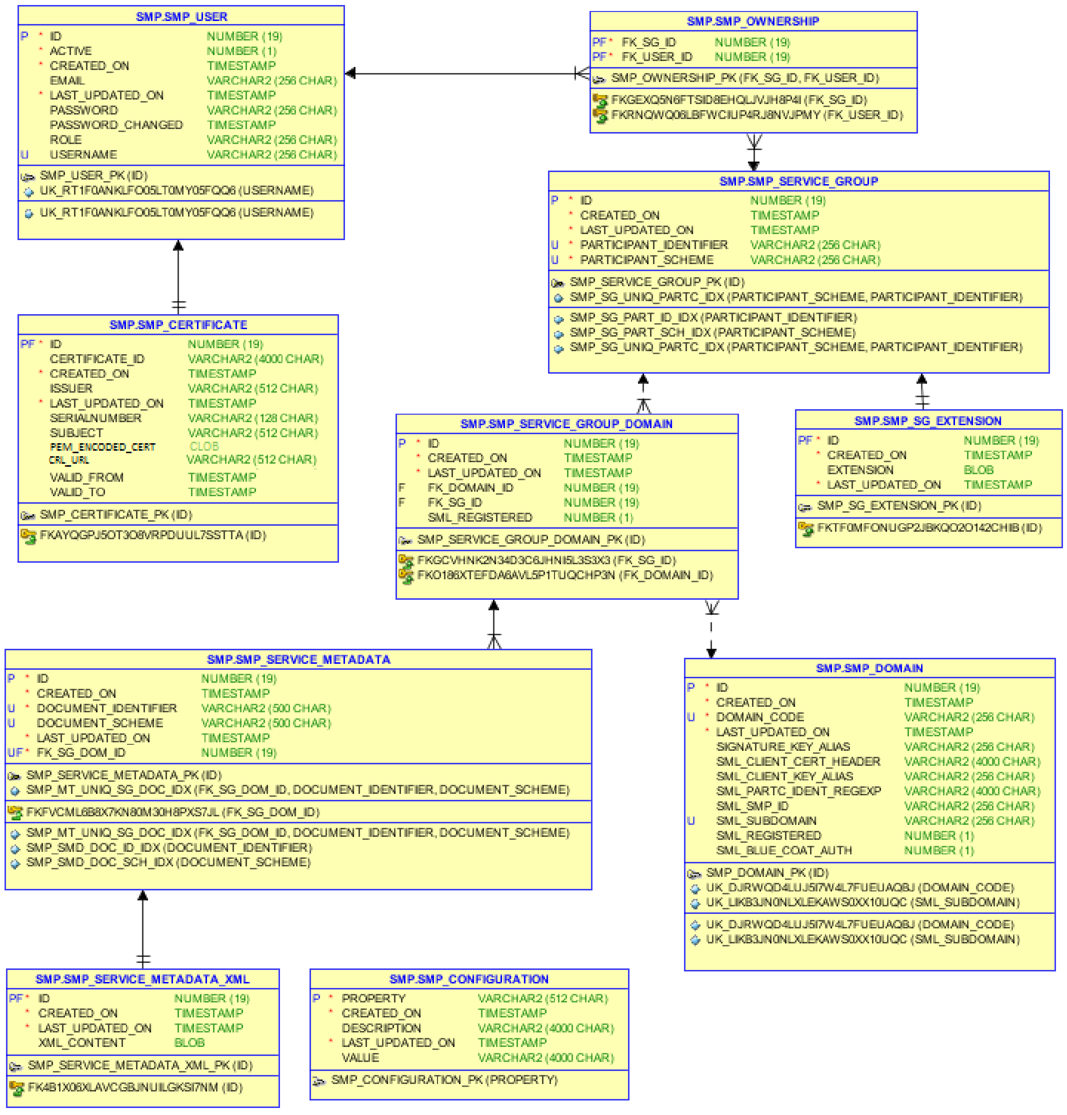

SMP.SMP_USER
P * ID NUMBER (19)
* ACTIVE NUMBER (1)
* CREATED_ON TIMESTAMP
EMAIL VARCHAR2 (256 CHAR)
* LAST_UPDATED_ON TIMESTAMP
PASSWORD VARCHAR2 (256 CHAR)
PASSWORD_CHANGED TIMESTAMP
ROLE VARCHAR2 (256 CHAR)
U USERNAME VARCHAR2 (256 CHAR)
SMP_USER_PK (ID)
UK_RT1F0ANKLFO05LT0MY05FQQ6 (USERNAME)
UK_RT1F0ANKLFO05LT0MY05FQQ6 (USERNAME)
SMP.SMP_OWNERSHIP
PF * FK_SG_ID NUMBER (19)
PF * FK_USER_ID NUMBER (19)
SMP_OWNERSHIP_PK (FK_SG_ID, FK_USER_ID)
FKGEXQ5N6FTSID8EHQLJVJH8P4I (FK_SG_ID)
FKRNQWQ06LBFWCIUP4RJ8NVJPMY (FK_USER_ID)
SMP.SMP_SERVICE_GROUP
P * ID NUMBER (19)
* CREATED_ON TIMESTAMP
* LAST_UPDATED_ON TIMESTAMP
U * PARTICIPANT_IDENTIFIER VARCHAR2 (256 CHAR)
U * PARTICIPANT_SCHEME VARCHAR2 (256 CHAR)
SMP_SERVICE_GROUP_PK (ID)
SMP_SG_UNIQ_PARTC_IDX (PARTICIPANT_SCHEME, PARTICIPANT_IDENTIFIER)
SMP_SG_PART_ID_IDX (PARTICIPANT_IDENTIFIER)
SMP_SG_PART_SCH_IDX (PARTICIPANT_SCHEME)
SMP_SG_UNIQ_PARTC_IDX (PARTICIPANT_SCHEME, PARTICIPANT_IDENTIFIER)
SMP.SMP_CERTIFICATE
PF * ID NUMBER (19)
CERTIFICATE_ID VARCHAR2 (4000 CHAR)
* CREATED_ON TIMESTAMP
ISSUER VARCHAR2 (512 CHAR)
* LAST_UPDATED_ON TIMESTAMP
SERIALNUMBER VARCHAR2 (128 CHAR)
SUBJECT VARCHAR2 (512 CHAR)
PEM_ENCODED_CERT CLOB
CRL_URL VARCHAR2 (512 CHAR)
VALID_FROM TIMESTAMP
VALID_TO TIMESTAMP
SMP_CERTIFICATE_PK (ID)
FKAYQGPJ5OT3O8VRPDUUL7SSTTA (ID)
SMP.SMP_SERVICE_GROUP_DOMAIN
P * ID NUMBER (19)
* CREATED_ON TIMESTAMP
* LAST_UPDATED_ON TIMESTAMP
F FK_DOMAIN_ID NUMBER (19)
F FK_SG_ID NUMBER (19)
SML_REGISTERED NUMBER (1)
SMP_SERVICE_GROUP_DOMAIN_PK (ID)
FKGCVHNK2N34D3C6JHNI5L3S3X3 (FK_SG_ID)
FKO186XTEFDA6AVL5P1TUQCHP3N (FK_DOMAIN_ID)
SMP.SMP_SG_EXTENSION
PF * ID NUMBER (19)
* CREATED_ON TIMESTAMP
EXTENSION BLOB
* LAST_UPDATED_ON TIMESTAMP
SMP_SG_EXTENSION_PK (ID)
FKTF0MFONUGP2JBKQO2O142CHIB (ID)
SMP.SMP_SERVICE_METADATA
P * ID NUMBER (19)
* CREATED_ON TIMESTAMP
U * DOCUMENT_IDENTIFIER VARCHAR2 (500 CHAR)
U DOCUMENT_SCHEME VARCHAR2 (500 CHAR)
* LAST_UPDATED_ON TIMESTAMP
UF * FK_SG_DOM_ID NUMBER (19)
SMP_SERVICE_METADATA_PK (ID)
SMP_MT_UNIQ_SG_DOC_IDX (FK_SG_DOM_ID, DOCUMENT_IDENTIFIER, DOCUMENT_SCHEME)
FKFVCML6B8X7KN80M30H8PXS7JL (FK_SG_DOM_ID)
SMP_MT_UNIQ_SG_DOC_IDX (FK_SG_DOM_ID, DOCUMENT_IDENTIFIER, DOCUMENT_SCHEME)
SMP_SMD_DOC_ID_IDX (DOCUMENT_IDENTIFIER)
SMP_SMD_DOC_SCH_IDX (DOCUMENT_SCHEME)
SMP.SMP_DOMAIN
P * ID NUMBER (19)
* CREATED_ON TIMESTAMP
U * DOMAIN_CODE VARCHAR2 (256 CHAR)
* LAST_UPDATED_ON TIMESTAMP
SIGNATURE_KEY_ALIAS VARCHAR2 (256 CHAR)
SML_CLIENT_CERT_HEADER VARCHAR2 (4000 CHAR)
SML_CLIENT_KEY_ALIAS VARCHAR2 (256 CHAR)
SML_PARTC_IDENT_REGEXP VARCHAR2 (4000 CHAR)
SML_SMP_ID VARCHAR2 (256 CHAR)
U SML_SUBDOMAIN VARCHAR2 (256 CHAR)
SML_REGISTERED NUMBER (1)
SML_BLUE_COAT_AUTH NUMBER (1)
SMP_DOMAIN_PK (ID)
UK_DJRWQD4LUJ5I7W4L7FUEUAQBJ (DOMAIN_CODE)
UK_LIKB3JN0NLXLEKAWS0XX10UQC (SML_SUBDOMAIN)
UK_DJRWQD4LUJ5I7W4L7FUEUAQBJ (DOMAIN_CODE)
UK_LIKB3JN0NLXLEKAWS0XX10UQC (SML_SUBDOMAIN)
SMP.SMP_SERVICE_METADATA_XML
PF * ID NUMBER (19)
* CREATED_ON TIMESTAMP
* LAST_UPDATED_ON TIMESTAMP
XML_CONTENT BLOB
SMP_SERVICE_METADATA_XML_PK (ID)
FK4B1X06XLAVCGBJNUILGKSI7NM (ID)
SMP.SMP_CONFIGURATION
P * PROPERTY VARCHAR2 (512 CHAR)
* CREATED_ON TIMESTAMP
DESCRIPTION VARCHAR2 (4000 CHAR)
* LAST_UPDATED_ON TIMESTAMP
VALUE VARCHAR2 (4000 CHAR)
SMP_CONFIGURATION_PK (PROPERTY)

Figure 16 Database ERD diagram

Besides all the necessary metadata used by the SMP business logic, the database is also used to store XML documents in two dedicated table (oracle: blob, mysql: TEXT type). First document is an EXTENSION in the table SMP_SG_EXTENSION. The column extension contains custom XML data defined by a service group owner. Second is XML_CONTENT in a table SMP_SERVICE_METADATA_XML. The column contains the original xml from insert/update metadata request. Original metadata is stored because it could be electronically signed by Service group owner. Decomposing and composing Metadata could compromise the xml signature. When a user is querying for the metadata, the original xml is returned with a valid xml signature.

The Java data access layer is implemented within the *smp-server-library* module. *DataSource, EntityManager* and *TransactionManager* are configured and registered into Spring context in the *DatabaseConfig* class.

Java classes located in *eu.europa.ec.edelivery.smp.data.model* package define the Model with the use of JPA2 annotations. All model classes implement the *BaseEntity* interface. Separate @Embeddable classes are defined for all composite primary keys:

```
@Entity
```

```
@Table(name = "smp_service_group")
public class DBServiceGroup implements BaseEntity {
    @EmbeddedId
    @Override
    public DBServiceGroupId getId() { return serviceGroupId; }
/* . . . */
}
```

Listing 16 Part of sample JPA2 Model class with embedded composite PK

```
@Embeddable
public class DBServiceGroupId implements Serializable {
    @Column(name = "businessIdentifierScheme", nullable = false, length =
MAX_IDENTIFIER_SCHEME_LENGTH)
    public String getBusinessIdentifierScheme() { return participantIdScheme; }

    @Column(name = "businessIdentifier", nullable = false, length =
MAX_IDENTIFIER_VALUE_LENGTH)
    public String getBusinessIdentifier() { return participantIdValue; }
/* . . . */
}
```

Listing 17 Part of sample @Embeddable composite PK

All DAO classes located in the *eu.europa.ec.edelivery.smp.data.dao* package extend the *BaseDao* generic abstract class that already provides most common DAO operations (find, remove, etc.).

```
@Repository
public class ServiceGroupDao extends BaseDao<DBServiceGroup> {}
```

Listing 18 Sample of the simplest DAO that doesn't need to provide additional methods

```
public abstract class BaseDao<E extends BaseEntity> {
    @PersistenceContext
    protected EntityManager em;

    private final Class<E> entityClass;

    public BaseDao() {
        entityClass = (Class<E>)
GenericTypeResolver.resolveTypeArgument(getClass(), BaseDao.class);
    }

    public E find(Object primaryKey) {
        return em.find(entityClass, primaryKey);
    }
/* . . . */
}
```

Listing 19 Significant part of the generic BaseDao

4.4. Exception handling

Detailed functional description of all errors that might occur is presented in the Interface Control Document (cf. [REF4]). This section presents a generalized view on error groups and focuses on implementation perspective.

eDelivery SMP utilizes HTTP error codes according to the best RESTful recommendations, i.e. given codes are always returned for:

- **200 (OK)** or **201 (Created)** – successful responses (resource created, updated, retrieved or deleted)
- **4xx (Bad request)** – invalid or unauthenticated request
- **5xx (Server Error)** – SMP technical issue, could be related to configuration, internal networking, integration with BDMSL or DB, etc.

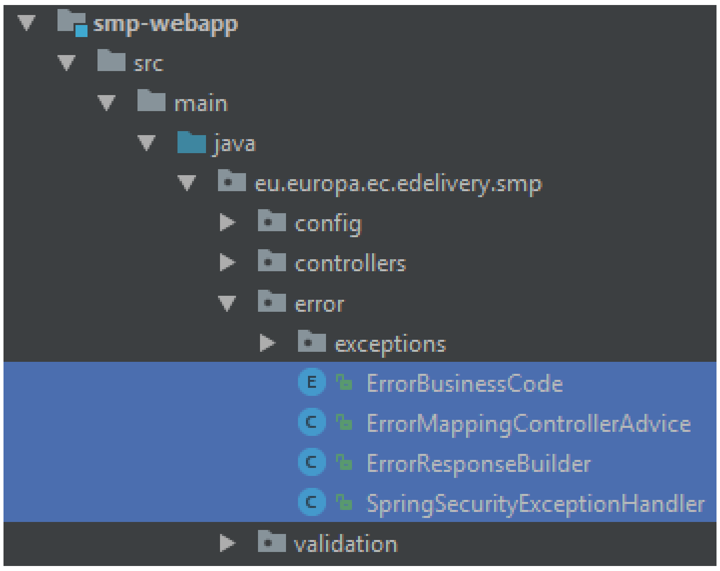
OASIS SMP specification (cf. [REF1]) does not specify error messages, thus eDelivery SMP introduces its own simple XSD with XML namespace: *ec:services:SMP:1.0*. This one describes the structure of error response messages as the sample below:

```
<ErrorResponse xmlns="ec : services:SMP:1.0">
   <BusinessCode>NOT_FOUND</BusinessCode>
   <ErrorDescription>ServiceMetadata not found, ServiceGroupID: 'x ::y',
DocumentID: 'a::b'</ErrorDescription>
   <ErrorUniqueId>2018-03-27T15 :07 :35.470CEST :d3ba543a-7233-4e69-9f34-
655e3998cb3c</ErrorUniqueId>
</ErrorResponse>
```

Listing 20 Sample error response

4.4.1. Error handling mechanism implementation

All classes for processing errors are located in package *eu.europa.ec.edelivery.smp.error*:



```
smp-webapp
  src
    main
      java
        eu.europa.ec.edelivery.smp
          config
          controllers
          error
            exceptions
            ErrorBusinessCode
            ErrorMappingControllerAdvice
            ErrorResponseBuilder
            SpringSecurityExceptionHandler
          validation
```

Figure 17 Classes implementing error handling mechanism

4.4.2. ErrorMappingControllerAdvice

All backend exceptions are mapped to REST responses within one single class registered in Spring context with *@RestControllerAdvice* and by its many handler-methods annotated with *@ExceptionHandler*. The class uses *ErrorResponseBuilder* and is responsible for:

- mapping exceptions to HTTP response codes and *ErrorBusinessCodes*
- logging user errors as WARN level and technical errors as ERROR level including *uniqueErrorId* for easier maintenance and debugging

Class declaration, sample handler-method (one of many) and internal re-used buildAndWarn method:

```
@RestControllerAdvice
public class ErrorMappingControllerAdvice {


    @ExceptionHandler(NotFoundException.class)
    public ResponseEntity handleNotFoundException(NotFoundException ex) {
        return buildAndWarn(NOT_FOUND, ErrorBusinessCode.NOT_FOUND,
ex.getMessage(), ex);
    }


/* . . . */
    private ResponseEntity buildAndWarn(HttpStatus status, ErrorBusinessCode
businessCode, String msg, Exception exception) { /* . . . */ }
}
```

Listing 21 Essential parts of ErrorMappingControllerAdvice class

4.4.3. *ErrorResponseBuilder*

ErrorResponseBuilder implementing builder pattern is responsible for building Spring's *ResponseEntity*, based on provided HTTP status code, *ErrorBusinessCode* and text message. Produced response not only is compliant with introduced dedicated XSD, but contains a *uniqueErrorId* that in future problem investigation can be easily found out in log files once user provides error message details.

Every *uniqueErrorId* is built out of:

- Timestamp – this information facilitates support and development by specifying when the error occurred and in which rolled log file more details can be found.
- UUID – helps in uniquely locating the error stack trace.

```
2018-03-27T15:07:35.470CEST:d3ba543a-7233-4e69-9f34-655e3998cb3c
```

Listing 22 Sample uniqueErrorId built out of timestamp and UUID

4.4.4. *ErrorBusinessCode*

ErrorBusinessCode is a simple *Enum* with given values, used by other error-handling classes:

Business error code	Description
XSD_INVALID	Bad request, XML document provided by the user does not pass schema validation
WRONG_FIELD	Bad request, one of the request fields is wrong, e.g. specified Domain does not exist.
OUT_OF_RANGE	Bad request, e.g. specified dates from-to are overlapped.
FORMAT_ERROR	Bad request, e.g. provided identifier format does not comply to OASIS SMP specifications (cf. [REF1])
UNAUTHORIZED	Unauthorized (HTTP 401), the user has no permission to access requested resource.
NOT_FOUND	Bad request, the requested resource does not exist (GET or DELETE).
USER_NOT_FOUND	Bad request, e.g. the newly created ServiceGroup cannot be owned by a user that does not exist.
TECHNICAL	Technical problem on SMP or infrastructure side (BDMSL integration, database etc.). This error is always returned with HTTP 500 "Internal

	Server Error" code. The specific cause of this error is not communicated in the response since Exceptions' messages might eventually reveal sensitive information.

4.4.5. SpringSecurityExceptionHandler

SpringSecurityExceptionHandler is a glue code that allows exceptions thrown by SpringSecurity to be processed by a common exception-handling mechanism. As a result, all security error responses follow the same pattern than other error responses.

SpringSecurity is implemented as a filter chain at the very beginning of the processing of HTTP requests.

5. Configuration

SMP configuration (database, keystore, authentication type …) is placed in the property file **smp.config.properties**. File with default values is already included in deployment war package. To override custom values the copy of **smp.config.properties** with updated values must be placed in the application server classpath. More details on configuring classpath can be found in the Administration Guide (cf. [REF3]) and in the §5.1 – "Environment specific configuration".

When the SMP is used in multi-tenancy as described in chapter §3.3 – "Domain Multitenancy", the configuration properties for domain (SMP ID, BDMSL authentication data) are located in database table: SMP_DOMAIN. One record represents one domain, columns represent configuration parameters which are applied for that specific domain. More details on domain configuring can be found in the Administration Guide (cf. [REF3])

5.1. Environment specific configuration

Detailed configuration steps for Windows and UNIX systems are covered in the SMP Administration Guide [REF3]. This section is focused explaining the motivation behind particular configuration rather than configuration steps themselves.

5.1.1. WebLogic

Classpath:

The SMP requires configuration file: **smp.config.properties** to be placed in the classpath. On weblogic server custom classpath folder (e.g. /conf_dir_path) can be set by modifying CLASSPATH variable in scripts setDomainEnv.sh:

```
EXPORT CLASSPATH="$CLASSPATH${CLASSPATHSEP}/conf_dir_path"
```

Listing 23 Adding SMP configuration dir to classpath

Authentication:

WebLogic by default validates username/password (*BasicAuth*) credentials if such are present in any incoming request. Because SMP handles *BasicAuth* with SpringSecurity this feature must be turned off. This is achieved by changing **enforce-valid-basic-auth-credentials** property in **config.xml** file to **false**.

5.1.2. Tomcat

Classpath:

The SMP requires configuration file: **smp.config.properties** to be placed in the classpath. On tomcat server custom classpath folder (e.g. /conf_dir_path) can be set by modifying the starting scripts in the same way as for WebLogic, or by adding this entry in context.xml file:

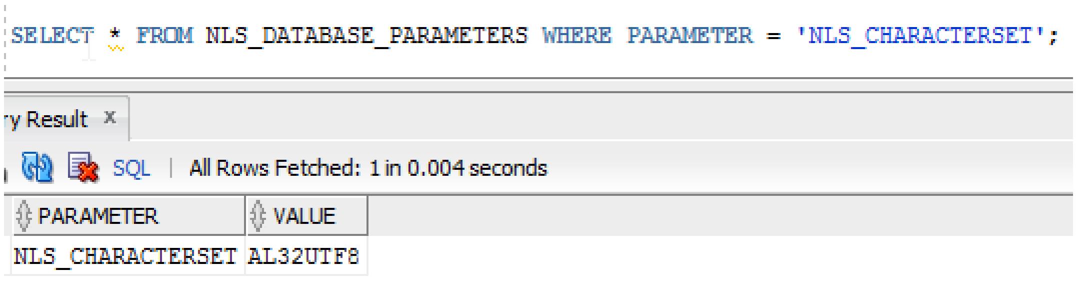
```
<Resources className="org.apache.catalina.webresources.StandardRoot"
cachingAllowed="true" cacheMaxSize="100000" >

    <PreResources className="org.apache.catalina.webresources.DirResourceSet"
```

```
        base="/conf_dir_path"
        internalPath="/"
        webAppMount="/WEB-INF/classes" />
</Resources>
```

Listing 24 Sample part of Tomcat's context.xml file presenting how to include configuration file into classpath

5.1.3. Oracle

NLS_CHARACTERSET must be set to AL32UTF8, otherwise SMP will face issues with non-ASCII characters.



```
SELECT * FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER = 'NLS_CHARACTERSET';
```

ry Result ✕

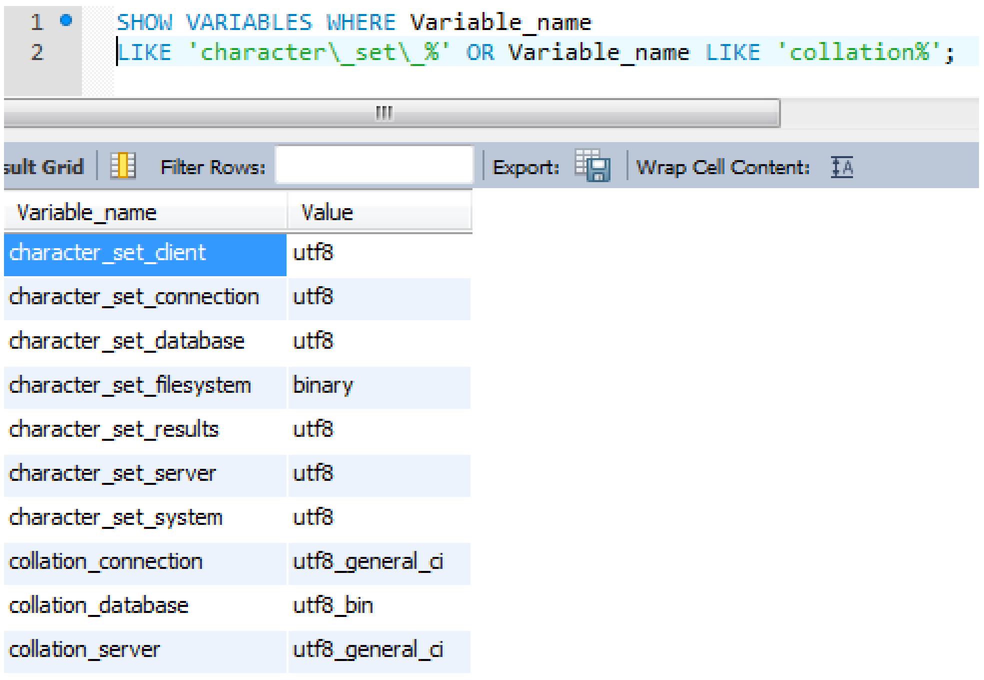
SQL | All Rows Fetched: 1 in 0.004 seconds

PARAMETER	VALUE
NLS_CHARACTERSET	AL32UTF8

Figure 18 Oracle NLS_CHARACTERSET must be set to AL32UTF8

5.1.4. MySql

Character set, collation and especially JDBC connection protocol encoding – all must be set to UTF-8, otherwise SMP will face issues with non-ASCII characters.



```
SHOW VARIABLES WHERE Variable_name
LIKE 'character\_set\_%' OR Variable_name LIKE 'collation%';
```

sult Grid | Filter Rows: | Export: | Wrap Cell Content:

Variable_name	Value
character_set_client	utf8
character_set_connection	utf8
character_set_database	utf8
character_set_filesystem	binary
character_set_results	utf8
character_set_server	utf8
character_set_system	utf8
collation_connection	utf8_general_ci
collation_database	utf8_bin
collation_server	utf8_general_ci

Figure 19 MySQL character encoding must be set to UTF8

6. Security

The SMP is secured with the SpringSecurity. The spring security configuration is executed at the eDelivery startup in the following classes:

- ***WSSecurityConfigurerAdapter.java***: class that handles the webservice endpoint security configuration,
- ***UISecurityConfigurerAdapter.java***: class that handles the UI endpoint security configuration,
- ***SMPCasConfigurer.java***: class that handles the UI Cas configuration.

6.1. Authentication

The Authentication Manager (id = smpAuthenticationManager) utilizes two Authentication One handles basic username/ password authentication and the second is SpringSecurity implementation *PreAuthenticatedAuthenticationProvider* class configured to handle *X509Certificate* and *BlueCoat* authentication. The pre-authenticated scenarios take precedence over basic authentication. That means if a client provided a valid certificate and also valid username and password, then he is logged in using his certificate and username/password is ignored.

6.1.1. Username and password authentication (Basic Authentication forUI)

Standard SpringSecurity mechanism is used to verify username and BCrypt hashed passwords using the SMPAuthenticationProvider. Username/Password authentication can be used for the UI authentication.

6.1.2. Access token authentication (Basic Authentication for web-services)

eDelivery SMP uses different credentials for UI and for WebService authentication.
The access token is randomly generated access token id and access token value. Together they are used as HTTP basic authentication when invoking the web-services.

6.1.3. Client certificate authentication

Client Certificate authentication can be used only for authentication when invoking the REST API services. The purpose of the certificate authentication is to support mutual 2-way TLS authentication for machine-to-machine integration.

SMP supports two types of Client Certificate authentications: X509 certificate authentication and Authentication behind Reverse Proxy. Both scenarios are performed in 2 steps:

1. Certificate details are extracted to the eDelivery-specific text format. This step is handled by two custom filters: *x509AuthFilter* and *blueCoatReverseProxyAuthFilter*, separately for both scenarios.
2. *PreauthAuthProvider* verifies that if certificate-defined user exists in the database.

X509Certificate and Certificates HTTP Client-Cert header are validated with the following attributes:

- Valid from: if “current date” is smaller than “valid from” date, then authentication is rejected

- Valid to: if “current date” is greater than “certificates valid to” date, then authentication is rejected
- Revocation List: certificates are validated by CRL which is downloaded and cached till the CRL “valid to” date. CRL URL endpoint is defined in SMP_CERTIFICATE.CRL_URL column and is used for HTTP Client-Cert authentication and for X509Certificate authentication. If the CRL is not reachable, SMP silently ignores the CRL verification, if the configuration attribute “smp.certificate.crl.force” is set to false. If the attribute is set to true, then Client is not authenticated due to technical issues.
- Truststore: If the SMP truststore is not empty, then formatted issuer or subject is verified if it exists in the truststore. If none of the values exists in the truststore, then certificate authentication is rejected.

Users that are authenticated by certificate are stored in the SMP_USER table, together with users authenticated by password. The USERNAME value of certificate authenticated users is a string value created from parts of certificate distinguish name (DN) and serial number by the following pattern (eDelivery format):

```
CN={common name},O={organisation},C={country}:{16-digit-zero-padded-hex-serial}
```

e.g.:

```
CN=CEF eDelivery,O=European Commission,C=BE:000000000000c41f
```

Application distinguished certificate authenticated users from password-authenticated user by an empty PASSWORD column.

Most eDelivery projects supporting client certificate authentication, utilize the same client certificate text representation and BlueCoat Client-Cert HTTP header patterns. For this reason custom Java code responsible for client certificate authentication has been extracted and released within a separate JAR library; maven dependency gropuId/artifactId: eu.europa.ec.edelivery/edelivery-springsecurity-2-way-ssl-auth.

6.1.3.1. X509 certificate authentication

The client X509 certificate authentication uses server's (Tomcat or WebLogic) certificate authentication settings. After the request passes the server validation successfully, *x509AuthFilter* extract certificate details and then authentication proceeds in the way as described above.

The filter itself (class *EDeliveryX509AuthenticationFilter*) is a simple extension of SpringSecurity's *X509AuthenticationFilter* class, which is a ready-to-use implementation handling *java.security.cert.X509Certificate*.

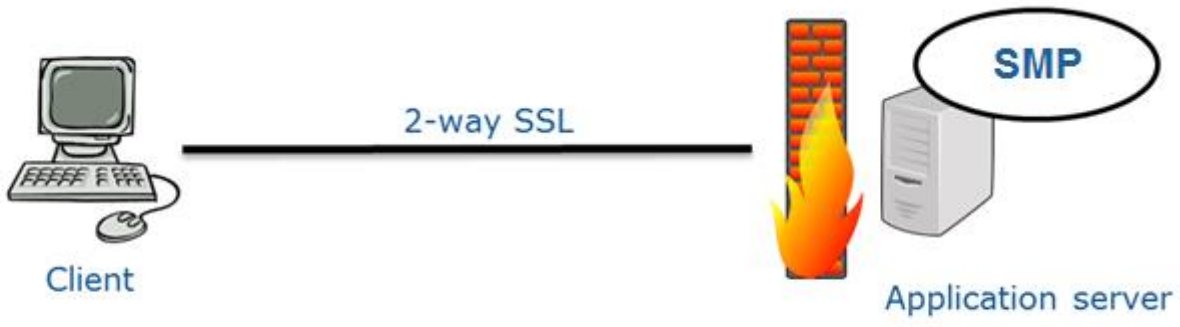


Figure 20 2-way-TLS scenario with truststore configured within J2EE container

6.1.3.2. Authentication behind Reverse Proxy

In this setup the basic certificate validation is configured in the BlueCoat reverse proxy. After certificate validation passed successfully, the BlueCoat reverse proxy adds a "Client-Cert" HTTP header and forwards the request to the SMP over HTTP(S). The spring filter *blueCoatReverseProxyAuthFilter* extracts the header, converts it from Bluecoat's to the eDelivery format specified above and then authentication proceeds in the way as described above.

The filter itself (class *BlueCoatAuthenticationFilter*) is based on the SpringSecurity's *RequestHeaderAuthenticationFilter*, dedicated for similar scenarios.

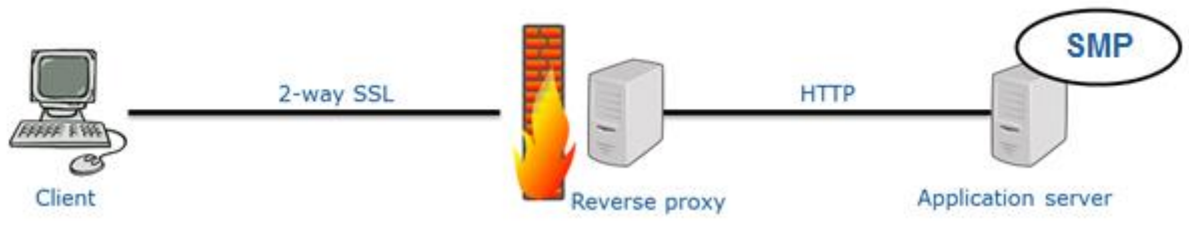
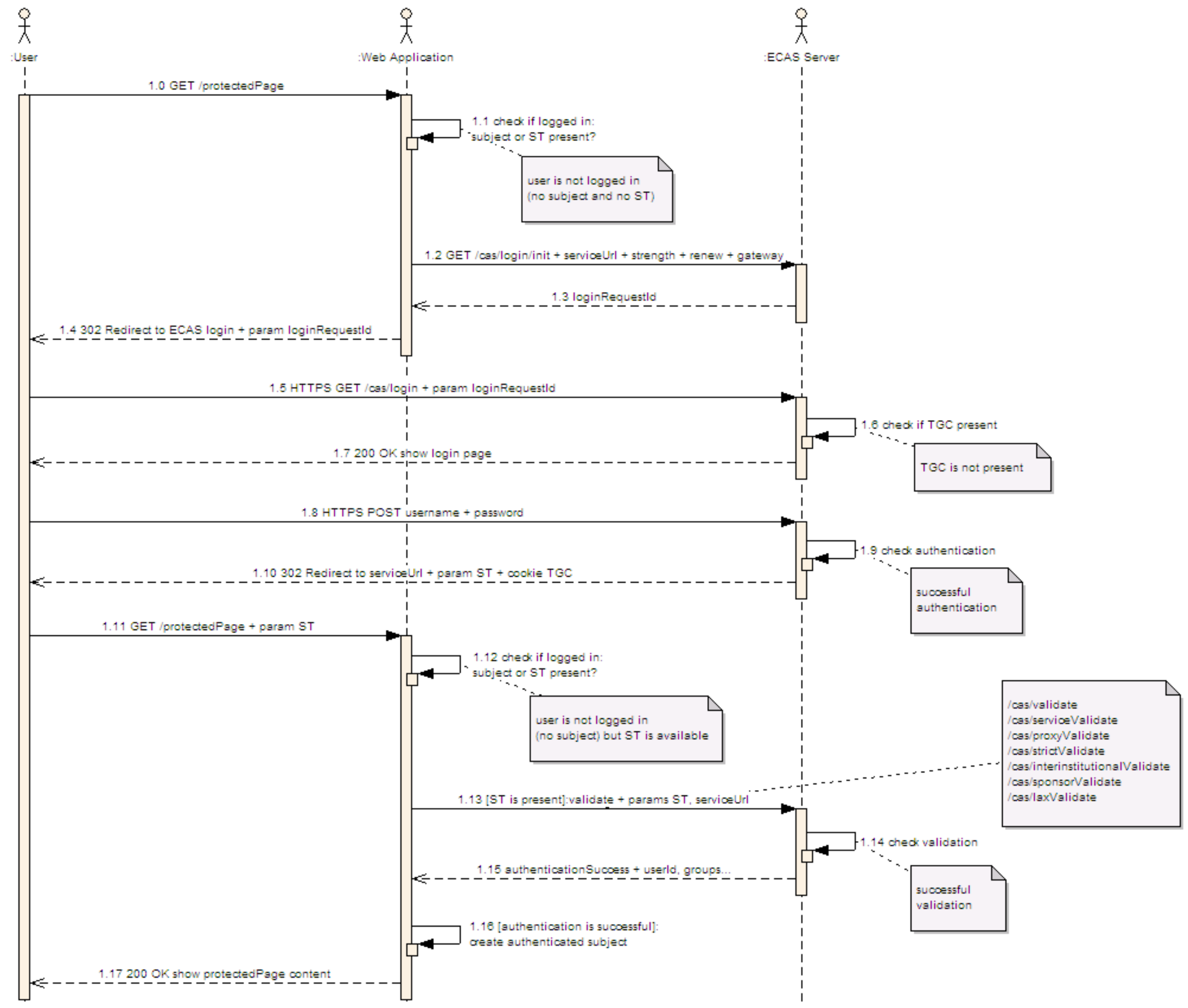


Figure 21 2-way-SSL scenario with BlueCoat reverse proxy

6.1.4. SSO Central Authentication service with EU-LOGIN

CAS authentication can be used only for the UI authentication, and it was made with intention to integrate with ECAS also called EU-Login. ECAS is based on the Central Authentication Service (CAS) version 2 developed at Yale University1. It is an authentication service to protect Web-based applications. SMP was tested only with ECAS, but it should also work with any CAS 2.0 implementation,


:User
:Web Application
:ECAS Server
1.0 GET /protectedPage
1.1 check if logged in: subject or ST present?
user is not logged in (no subject and no ST)
1.2 GET /cas/login/init + serviceUrl + strength + renew + gateway
1.3 loginRequestId
1.4 302 Redirect to ECAS login + param loginRequestId
1.5 HTTPS GET /cas/login + param loginRequestId
1.6 check if TGC present
TGC is not present
1.7 200 OK show login page
1.8 HTTPS POST username + password
1.9 check authentication
successful authentication
1.10 302 Redirect to serviceUrl + param ST + cookie TGC
1.11 GET /protectedPage + param ST
1.12 check if logged in: subject or ST present?
user is not logged in (no subject) but ST is available
/cas/validate
/cas/serviceValidate
/cas/proxyValidate
/cas/strictValidate
/cas/interinstitutionalValidate
/cas/sponsorValidate
/cas/laxValidate
1.13 [ST is present]:validate + params ST, serviceUrl
1.14 check validation
successful validation
1.15 authenticationSuccess + userId, groups...
1.16 [authentication is successful]: create authenticated subject
1.17 200 OK show protectedPage content

When the SMP does not find a *service ticket* granting access it redirects to *EU login* page for user authentication. After user is authenticates on EU-login, the response redirects page back to SMP UI page with granting ticket. SMP validates ticket with ECAS. If validation is successful the SMP authorize access to the user according to user authorization defined on SMP user configuration.

6.2. Authorization

6.2.1. Authorities

Authorities in SMP are organized into a two-dimensional space, with Roles as first dimension and ServiceGroup Ownership as the second one.

6.2.1.1. Roles

Roles are documented with more details in ICD (cf. [REF4]). The table below explains their meaning from the implementation perspective:

Role alias	Description
ROLE_ANONYMOUS	Any user that hasn't provided any authentication details
ROLE_SERVICE_GROUP_ADMIN	Any authenticated user that exists in the database. Such user is supposed to own at least one ServiceGroup on which he is allowed to perform some administrative actions.
ROLE_SMP_ADMIN	This user is allowed to perform every administrative action.
ROLE_SYSTEM_ADMIN	Role for UI enables administration of domains and users.

ICD mentions "System Admin" role, but it's rather a sysadmin, not the business role to be considered in SMP source code.

6.2.1.2. ServiceGroup Ownership

User with ROLE_SERVICE_GROUP_ADMIN must own at least one ServiceGroup. On the owned ServiceGroup he has permission to manipulate (add, delete, update) the ServiceMetadata.

6.2.2. Authorities execution

Authorities' verification is very flexible thanks to loading all granted authorities to the security context.

6.2.2.1. HTTP methods: GET/PUT/DELETE

The first level of verification is made on HTTP method level. GET is allowed to everybody, while all modifying actions are allowed only to authenticated users, which is configured in spring-security.xml file:

```
<intercept-url method="PUT" access=" ! isAnonymous()" pattern="/*"/>
<intercept-url method="DELETE" access=" ! isAnonymous()" pattern="/*"/>
```

6.2.2.2. Business object and action level

Once all granted authorities are present in the security context, they are validated at the business methods level with SpringSecurity's annotations and Spring Expression Language (SpEL):

```
@Secured("ROLE_SMP_ADMIN")
```

- action allowed only for SMP Admin, or:

```
@PreAuthorize("hasAnyAuthority('ROLE_SMP_ADMIN',
         @caseSensitivityNormalizer.normalizeParticipantId(#serviceGroupId))")
```

- action allowed either for SMP Admin or ServiceGroup admin owing the "serviceGroupId" provided as methods' parameter.

7. Quality

SMP quality is supervised by Code Reviews and Continuous Integration processes, which are out of the scope of this document. The quality measurement details presented below focus on technical and source-code point of view.

7.1. Unit tests

All utility classes that don't interact with many other classes, which are mostly responsible for conversions, mappings, etc., are unit tested with using Junit and Mockito libraries. Test class name pattern in this case is: *{testedClassName}Test.java*. Tests are run at application build time.

7.2. Integration tests

Service classes that combine multiple application modules and in most of the cases require database access are tested in classes with name pattern: *{testedClassName}IntegrationTest.java*. Tests are executed with JUnit library and configured Spring test context. Also database instance must be created and defined in maven project files with the following properties:

Property	Description
jdbc.driver	Database Configuration: Driver MySQL: - com.mysql.jdbc.Driver Oracle Database: - oracle.jdbc.OracleDriver
jdbc.url	Database Configuration: url - MySQL: jdbc:mysql://dbhost:dbport/smp_database - Oracle Database: jdbc:oracle:thin:@dbhost:dbport:smp_database or jdbc:oracle:thin:@dbhost:dbport/smp_service
jdbc.password	Database User/Password Configuration: User
jdbc.password	Database User/Password Configuration: Password
target-database	Target Database Backend type/Brand: For MySQL, use: MySQL For Oracle Database, use: Oracle
jdbc.read-connections.max	Database Configuration: Max Read Connection

Example:

```
<properties>
    <jdbc.driver>com.mysql.jdbc.Driver</jdbc.driver>
    <jdbc.url>jdbc:mysql://localhost/smp</jdbc.url>
    <jdbc.user>smp</jdbc.user>
    <jdbc.password>smp</jdbc.password>
    <target-database>MySQL</target-database>
    <jdbc.read-connections.max>10</jdbc.read-connections.max>
</properties>
```

7.3. SoapUI integration tests

All functionalities are covered with SoapUI integration tests that run REST requests against the SMP and in some cases access the database directly with SQL statements. The SoapUI project can be found in submodule *smp-soapui-tests\soapui\SMP4.0-Generic-soapui-project.xml* file. These tests are bound to maven build and can be activated at build time with maven profile *-Prun-soapui* switch.

7.4. Sonar source code statistics

Maven build is configured to collect standard Sonar code statistics (code test coverage, static code analysis, etc.). Apart from that, code test coverage is gathered also when running SoapUI tests. This requires manual install of Jacoco Agent in JRE with J2EE container where the SMP is deployed and pointing to this agent when running a build by adding these attributes to maven run:
-DjacocoRemotePort=65000 –DjacocoRemoteAddress.

Once build with SoapUI tests is done, statistics from all the sources are gathered by sonar plugin by running *mvn sonar:sonar* goal.

8. TECHNICAL REQUIREMENTS

This chapter describes the minimum and recommended system requirements to operate the SMP component.

8.1. Hardware

Type	Minimum	Recommended
Processor	1 CPU core	4 CPU core
Memory (RAM)	2GB	8GB or more
Disk space	5GB	Depends on usage

8.1.1. Recommended stack

Ubuntu 20.04 LTS 64 bits
Oracle Java EE 8
MySQL 5.6

8.1.2. Operating Systems

Any operating system that is compliant with the supported JVM.

8.1.3. Java Virtual Machines

Oracle Java SE JRE 8

8.1.4. Java Application Servers

Apache Tomcat 8.x
Oracle WebLogic Server 12c (12.1.2.0.0)

8.1.5. Databases

MySQL 5.6
Oracle Database 11g (11.2.0.4.0)

8.1.6. Web Browsers

n/a

9. List of Figures

10. LIST OF LISTINGS

11. Contact Information

eDelivery Support Team

By email: EC-EDELIVERY-SUPPORT@ec.europa.eu

Standard Service: 8am to 6pm (Normal EC working Days)