



## COMPETITIVENESS AND INNOVATION FRAMEWORK PROGRAMME ICT Policy Support Programme (ICT PSP)

### Towards pan-European recognition of electronic IDs (eIDs)

ICT PSP call identifier: ICT-PSP/2007/1

ICT PSP Theme/objective identifier: 1.2

### Project acronym: STORK

Project full title: Secure Identity Across Borders Linked

Grant agreement no.: 224993

## D5.8.3a Software Architecture Design

<b>Deliverable Id :</b>	<b>D5.8.3a</b>
<b>Deliverable Name :</b>	<b>D5.8.3a Software Architecture Design</b>
<b>Status :</b>	<b>For review</b>
<b>Dissemination Level :</b>	<b>Public</b>
<b>Due date of deliverable :</b>	<b>December 31<sup>st</sup> 2011</b>
<b>Actual submission date :</b>	<b>November 11<sup>th</sup> 2011</b>
<b>Work Package :</b>	<b>5.1</b>
<b>Organisation name of lead contractor for this deliverable :</b>	<b>ES-MAP</b>
<b>Author(s):</b>	<b>Diana Berbecaru, Eva Jorquera, Joaquin Alcalde-Moraño, Renato Portela, Wolfgang Bauer, Bernd Zwattendorfer, Jan Eichholz, Tim Schneider</b>
<b>Partner(s) contributing :</b>	<b>IT, PT, ES, AT, DE</b>

**Abstract:** This document describes the architecture of the systems that compose the common functionalities of the STORK platform. This description is made from various points of view, each described in a separate chapter. The relevant points of view are applied to each of the two systems: PEPS and MiddleWare (MW), this last one including the Virtual IDP.

Each of these systems is described in a separate chapter subdivided in subchapters.

As this document is an annex of D5.8.3 Technical Design, please refer to the abstract on the mother document.

## History

<i>Version</i>	<i>Date</i>	<i>Modification reason</i>	<i>Modified by</i>
0.1	7/10/2010	Initial version (same version as 5.8.2a)	Diana Berbecaru, Eva Jorquera, Joaquin Alcalde-Moraño, Tim Schneider, Renato Portela, Wolfgang Bauer, Bernd Zwattendorfer, Jan Eichholz, Jorge López
0.2	15/10/2011	Updated version	Ricardo da Silva Ferreira, Joaquín Alcalde-Moraño Jensen
Final 1.0	11/11/2011	Finalization	S. Koppius, A. v. Overeem, R. Wannee

Intermediate internal versions, e.g. for quality reviews, have been omitted.

## Table of contents

<b>HISTORY.....</b>	<b>2</b>
<b>TABLE OF CONTENTS .....</b>	<b>3</b>
<b>LIST OF FIGURES.....</b>	<b>5</b>
<b>LIST OF TABLES.....</b>	<b>6</b>
<b>EXECUTIVE SUMMARY .....</b>	<b>7</b>
<b>1 INTRODUCTION .....</b>	<b>8</b>
1.1 OBJECTIVE.....	8
1.2 CONTENTS OF THE DOCUMENT.....	8
1.3 SCOPE .....	8
1.4 METHODOLOGY.....	8
<b>2 PEPS.....</b>	<b>9</b>
2.1 SYSTEM CONTEXT.....	9
2.2 USE CASE VIEW AND OTHER REQUIREMENTS .....	10
2.2.1 USE CASE VIEW .....	10
2.2.2 NON FUNCTIONAL REQUIREMENTS (NFR) .....	11
2.3 LOGICAL VIEW .....	12
2.3.1 S-PEPS.....	12
2.3.2 C-PEPS .....	29
2.3.3 CONFIG FILES .....	51
2.3.4 ADMINISTRATION TASKS .....	52
2.3.5 ERROR HANDLING .....	52
<b>3 MIDDLEWARE .....</b>	<b>54</b>
3.1 SYSTEM CONTEXT.....	54
3.2 USE CASE VIEW AND OTHER REQUIREMENTS .....	54
3.2.1 USE CASE VIEW .....	55
3.2.2 NON FUNCTIONAL REQUIREMENTS.....	56
3.3 LOGICAL VIEW .....	56
3.3.1 USE CASE ANALYSIS .....	58
3.4 DEPLOYMENT VIEW.....	73
3.5 DATA VIEW.....	74
3.5.1 CLASS VIDPSTARTAUTHREQUEST .....	75
3.5.2 VIDPGETAUTHDATAREQUEST.....	76
3.5.3 VIDPGETAUTHDATARESPONSE .....	76
3.5.4 VIDPSTARTAUTHRESPONSE.....	76
3.5.5 SPWARESTARTAUTHREQUEST .....	77

3.5.6	SPWAREGETAUTHDATAREQUEST .....	77
3.5.7	SPWAREGETAUTHDATARESPONSE.....	77
3.5.8	SPWARESTARTAUTHRESPONSE .....	77

## List of figures

<i>Figure 1: RUP 4+1 view model .....</i>	<i>8</i>
<i>Figure 2: System Context Diagram.....</i>	<i>9</i>
<i>Figure 3: Use case view .....</i>	<i>10</i>
<i>Figure 4: Sequence diagram Prerequisite for SP without SAML capacities .....</i>	<i>13</i>
<i>Figure 5: Sequence diagram Authentication in S-PEPS.....</i>	<i>17</i>
<i>Figure 6: Sequence diagram Certificate Validation in S-PEPS.....</i>	<i>26</i>
<i>Figure 7: Sequence diagram Authentication in C-PEPS .....</i>	<i>30</i>
<i>Figure 8: Sequence diagram Certificate Validation in C-PEPS.....</i>	<i>45</i>
<i>Figure 9: MW System Context.....</i>	<i>54</i>
<i>Figure 10: Use Case Diagram .....</i>	<i>55</i>
<i>Figure 11: STORK-MW Components.....</i>	<i>56</i>
<i>Figure 12: UC-AU-P-MOA-ID .....</i>	<i>59</i>
<i>Figure 13: UC-AU-P-eIdService.....</i>	<i>62</i>
<i>Figure 14: UC-AU-MP.....</i>	<i>65</i>
<i>Figure 15: UC-AU-M-MOA-ID .....</i>	<i>67</i>
<i>Figure 16: UC-AU-M-eIdService.....</i>	<i>70</i>
<i>Figure 17 -UC-AU-SPEPS.....</i>	<i>72</i>
<i>Figure 18 Deployment Options for MARS architecture.....</i>	<i>74</i>
<i>Figure 19: Data model.....</i>	<i>75</i>

## List of tables

<i>Table 1 – User requirements .....</i>	<i>11</i>
<i>Table 2 – Evolution requirements .....</i>	<i>12</i>
<i>Table 3 – Description sequence for SP without SAML capacities .....</i>	<i>16</i>
<i>Table 4 – Description sequence Authentication in S-PEPS .....</i>	<i>25</i>
<i>Table 5 – Description sequence Certificate Verification in S-PEPS.....</i>	<i>29</i>
<i>Table 6 –Description sequence Authentication in C-PEPS.....</i>	<i>43</i>
<i>Table 7 – Description sequence Certificate Validation in C-PEPS .....</i>	<i>51</i>
<i>Table 8 – Configuration files in PEPS .....</i>	<i>52</i>
<i>Table 9 – PEPS Administrative tasks .....</i>	<i>52</i>
<i>Table 10 – Error Catalogue .....</i>	<i>53</i>
<i>Table 11 – Stork MW components description.....</i>	<i>58</i>
<i>Table 12 – Authentication process for AT citizens accessing SP of a PEPS country.....</i>	<i>61</i>
<i>Table 13 –Authentication process for DE citizens accessing SP of a PEPS country.....</i>	<i>63</i>
<i>Table 14 –Authentication process for citizens of PEPS country accessing SP of a MW country .</i>	<i>66</i>
<i>Table 15 –Authentication process for citizens of PEPS country accessing AT-SP .....</i>	<i>69</i>
<i>Table 16 –Authentication process for citizens of PEPS country accessing DE SP.....</i>	<i>71</i>
<i>Table 17 –Authentication process for V-IDP with MARS technology.....</i>	<i>73</i>
<i>Table 18 – Attributes of VIDPStartAuthRequest .....</i>	<i>76</i>
<i>Table 19 – Attributes of VIDPGetAuthDataRequest .....</i>	<i>76</i>
<i>Table 19 – Attributes of VIDPGetAuthDataResponse.....</i>	<i>76</i>
<i>Table 21 –Attributes of VIDPStartAuthResponse.....</i>	<i>76</i>
<i>Table 22 –Attributes of SPWareStartAuthRequest .....</i>	<i>77</i>
<i>Table 23 – Attributes of SPWareGetAuthDataRequest .....</i>	<i>77</i>
<i>Table 24 – Attributes of SPWareGetAuthDataResponse.....</i>	<i>77</i>
<i>Table 25 – Attributes of SPWareStartAuthResponse.....</i>	<i>78</i>

## Executive summary

This document describes the architecture of the systems that compose the common functionalities of the STORK platform. This description is made from various points of view, each described in a separate chapter. The relevant points of view are applied to each of the two systems: PEPS and MiddleWare (MW), this last one including the Virtual IDP.

Each of these systems is described in a separate chapter subdivided in subchapters.

In next documents (D5.8.3c and D5.8.3e) views by components and classes will be offered.

This document is an annex of D5.8.3 Technical design. Please refer to the executive summary in that document.

# 1 Introduction

This document is an annex of D5.8.3 Technical design. So please also refer to the introduction of that document, especially for risk management, acceptance criteria and glossary.

## 1.1 Objective

This document describes the architecture of the systems that compose the common functionalities of the Stork platform. This description is made from various points of view, each described in a separate chapter. The relevant points of view are applied to each of the two systems: Pan European Proxy Services (PEPS) and MiddleWare (MW), this last one including the Virtual ID Provider (V-IDP).

## 1.2 Contents of the document

This document describes, after this first Introduction chapter, in the two systems, each in one chapter. Each of these chapters starts with a general context description, zooming in, in the next subchapter with the use cases and other requirements.

After these general overviews, the description gets to more detail in the next subchapters describing each of them one of the views which is considered relevant by the architecture designers.

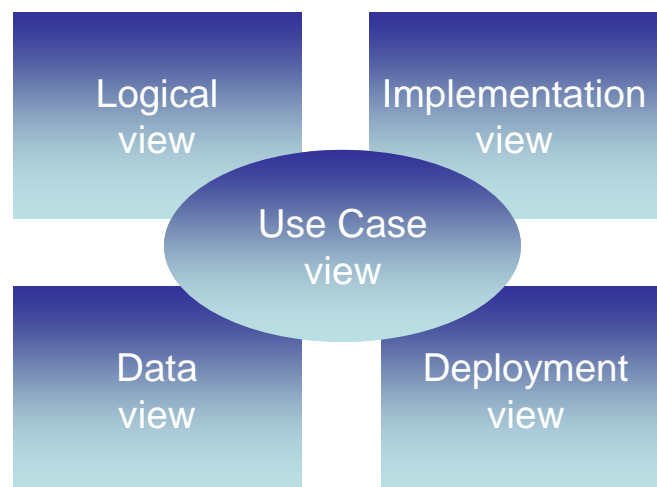
## 1.3 Scope

This document only describes the common functionalities of the Stork Platform. Specific functionalities, organisational and infrastructure aspects are to be determined by each member state.

## 1.4 Methodology

The methodology is based on the *RUP (Rational Unified Process) 4+1* view model. This model considers 5 views as normally sufficient to describe a system, the first one being the use case view. Nevertheless, in different publications, different views are described for the other 4 views, although all agree on the most important one: the logical view.

In this logical view the system is divided in subsystems, and for each subsystem the different business processes (Authentication and Certificate Validation) are described.

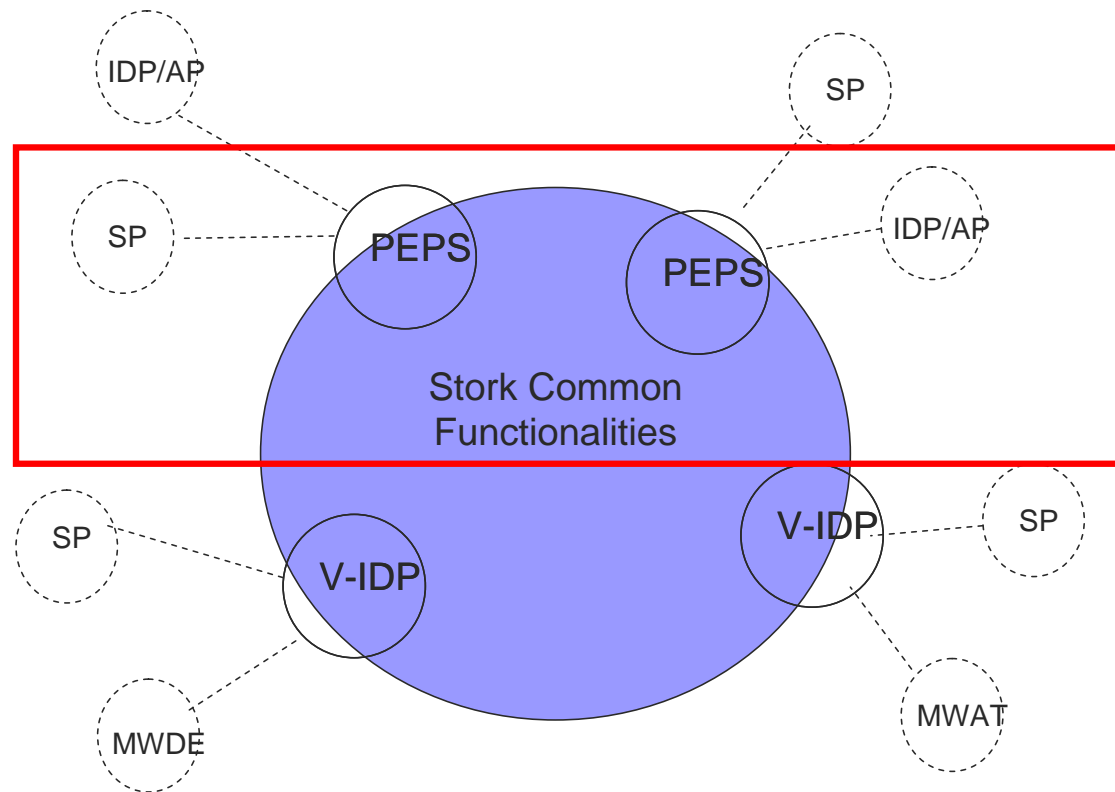


**Figure 1: RUP 4+1 view model**



## 2 PEPS

### 2.1 System Context



*Figure 2: System Context Diagram*

In each instance of a PEPS there are 2 roles: the one that attends to SP requests, and the one in the country which issued the ID of the citizen will use. Please note that the nationality of the ID issuer need not be the same as the nationality of the citizen.

For each received request, the first one forwards this request to his colleague PEPS or V-IDP, and the second (role of the) PEPS resolves the requests received from his colleague PEPS or V-IDP.

Each PEPS may include the functionalities which are specific to its member state, which are typically the interfaces with the local Id providers and attribute providers. On the other hand, the interface with Service providers (SPs) may also be different from one country to the other one.

But the communication between PEPSes and V-IDP, and the common functionalities are standard. This blue part of the above diagram, within the red rectangle is object of description in the PEPS chapter of this document. The communication between a PEPS and a V-IDP is the same as between 2 PEPSes.

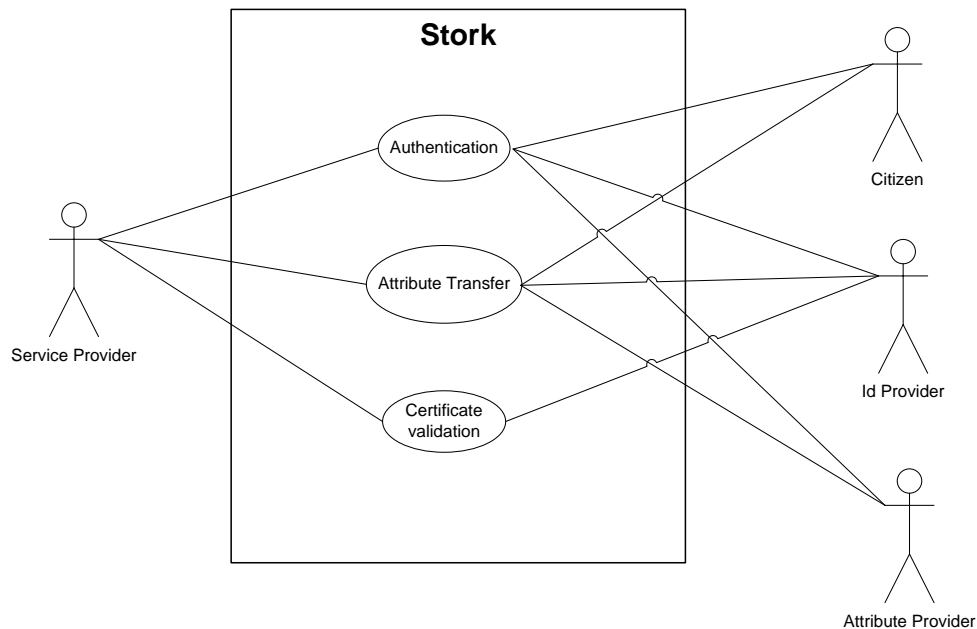
The other important chapter in this document describes the V-IDP, which is in charge to translate the common communication to the communication agreed between the MW countries.

This system also includes both roles, attending requests from their SPs and forwarding them to the corresponding PEPS, as well as attending requests from PEPSes.

## 2.2 Use case view and other requirements

### 2.2.1 Use case view

The following diagram shows the system level use-cases offered by the STORK system.



*Figure 3: Use case view*

The functionality of STORK is defined by the following processes.

1. **Authentication and Identification** is the process of verifying the identity of a particular user. This is achieved by asking for information that proves his identity. As a result of this process, the user is allowed to access privileged data. Usually this process ends with a fully identified user, which means that his identifier is transferred to the service provider (SP), and this SP recognises this user as a known customer, student, partner, or whatever relationship this person may have with the SP.

Nevertheless, other service providers or applications may exist that have fewer requirements; they allow access to privileged data without fully identifying the user. In the pilots of the STORK project we have examples of the access to university library, which is reserved to students, or the access to several rooms of Saferchat, which is restricted to people of a certain range of age, or of a certain gender. In both these cases we should call this process authentication, even though it doesn't end with a fully identified user. Other examples exist of authentication processes that don't end up with a fully identified user against SP. The user is identified in their country but only the data required to access is sent to the SP.

Please note that **attribute provisioning**, which is retrieval of attributes from an attribute provider, may be included in the basic process of authentication; depending on the requested attributes and the organisation of available attributes in each Member State.

2. **Attribute Transfer** is the process that allows a service provider to access additional attributes from an attribute provider (AP), other than those required for the basic

authentication. This process is initiated with a fully identified user, but his eID is not recognised as a known customer at the SP.

The attributes that can be requested are any combination of data-types recognised by STORK, and the STORK platform will do whatever it can, to retrieve the requested data, and, if the user allows so, pass the data to the service-provider that requests this info. This user-consent will be implemented in accordance with the legal data protection requirements of each Member State.

The implementation of authentication and attribute transfer as two consecutive processes might require the user to identify himself twice to his IDP, give twice his consent to the transfer of data, etc., so such a construction should be considered as improvable. But several applications exist, which contain such an implementation, so their support within Stork is unavoidable.

Nevertheless, in the actual definition of the project, Attribute Transfer is not considered within the scope of the pilots' requirements.

**Please note that Attribute Transfer is not required for the pilots, so will be left out for the moment.**

3. **Certificate validation** handles the scenario, where the service provider (SP) needs to verify a user-created digital signature. Whereas the signature validation is out of scope, STORK offers the functionality of validating the X509 certificate used for producing the signature.
4. Another process corresponds to the **registration of a user** for a particular service provided by a Service Provided. This process is, within Stork, exactly the same as authentication, except for the requested attributes: these will usually be more than in case of authentication of a known user.

## 2.2.2 Non Functional requirements (NFR)

Non functional requirements for the pilots are very limited, as the use of the system will be very limited. The NFR included in this chapter will be filled in when applicable, which will be in the final situation when the Stork eID interoperability platform is widely used.

### 2.2.2.1 NFR: User Requirements

Name	Description
<i>Availability</i>	Although the pilots don't have special requirements for availability, the system should be designed for high availability, and implemented that way
<i>Capacity</i>	The usage of the system, due to the requests from the pilots will be limited to thousands of transactions a year. Although equipments where these systems will be installed may be configured for such a limited usage, upgrades must be foreseen, and easy (quick) to implement.
<i>Reliability</i>	Stored data cannot be corrupted by defective code, concurrent access, or unexpected process termination.
<i>Performance</i>	A response time of 5 seconds is acceptable for each of the interactions

*Table 1 – User requirements*

Name	Description
<i>Scalability</i>	The system must be designed to scale well with <ul style="list-style-type: none"> <li>more countries (we can expect up to some 30)</li> </ul>

	<ul style="list-style-type: none"> <li>• more IdP's</li> <li>• very many SP's</li> </ul>
<i>Flexibility</i>	The system must allow for personalisation in each of the member states. This applies of course to the user interface, but also to available data, etc.
<i>Portability</i>	The common code should work <ul style="list-style-type: none"> <li>• with Tomcat, JBoss and Glassfish</li> <li>• under Linux (Ubuntu) and Windows 2003 server</li> </ul>
<i>Reusability</i>	Components of the system must be created to function and integrate within more than one environment
<i>Extensibility</i>	The system must be prepared
<i>Maintainability</i>	Although the project has a limited duration, an extension should be foreseen, so the code should obey to normal maintainability criteria.

**Table 2 – Evolution requirements**

## 2.3 Logical view

The main goal of the logical view is the decomposition of the system into subsystems. This can be done by component and/or class diagrams, showing the architecturally important components and their relationships.

The sequence diagrams show the sequence of messages passed between objects using a vertical timeline.

### 2.3.1 S-PEPS

#### 2.3.1.1 Authentication process

The authentication process is carried out using the citizen's Web Browser as a gateway for every message that needs to be exchanged between two STORK entities. Thereby, the authentication request a SP has to send to the S-PEPS will be performed through the citizen's Web Browser. In the same way, the S-PEPS will redirect (if needed) the authentication request (as a SAML AuthnRequest) to the C-PEPS through the citizen's Web Browser as well.

Some European regulations on privacy and personal data protection have obliged STORK to use this mechanism for the sending of user information amongst the members of the STORK infrastructure. This requirement is supported by the OASIS SAML HTTP redirect/POST profile.

The authentication process begins when the SP sends a SAML AuthRequest to the S-PEPS (Sequence diagram AU). If the SP doesn't have the ability to generate SAML messages then it can ask for a SAML AuthRequest to the S-PEPS (Sequence diagram Prerequisite for SP without SAML capacities).

### 2.3.1.1.1 Sequence diagram Prerequisite for SP without SAML capacities

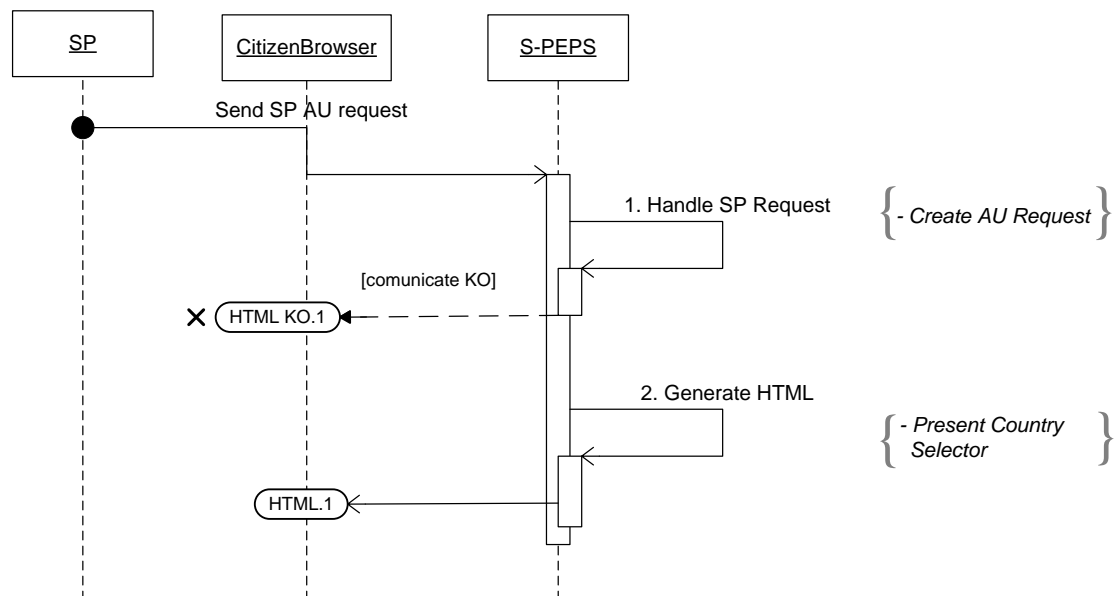
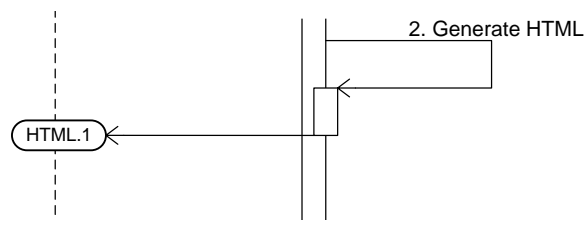
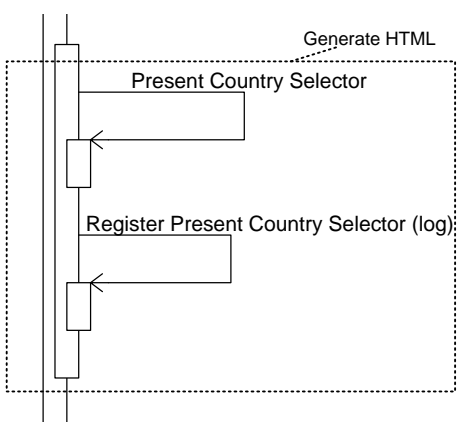


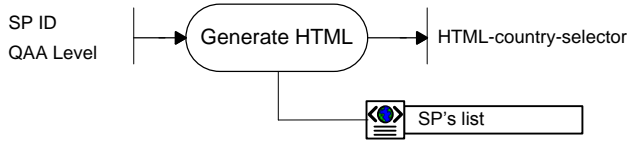
Figure 4: Sequence diagram Prerequisite for SP without SAML capacities

### 2.3.1.1.2 Description

Message sequences (interactions)		Description
1	Handle SP Request	<b>Description</b> The objective of this activity is to check the origin of the request and decide if the request, in this first step, is accepted or not.
		<b>Sequence Diagram</b> <pre> sequenceDiagram     participant SP     participant CitizenBrowser     participant S-PEPS      SP-&gt;&gt;S-PEPS: Send SP AU request     activate S-PEPS     S-PEPS-&gt;&gt;S-PEPS: 1. Handle SP Request     S-PEPS--&gt;&gt;CitizenBrowser: [communicate KO]     activate CitizenBrowser     CitizenBrowser-&gt;&gt;CitizenBrowser: HTML KO.1     deactivate CitizenBrowser     deactivate S-PEPS   </pre>

		<p><b>Detailed Sequence Diagram</b></p> <pre> sequenceDiagram     participant CB as Citizen-Browser     participant S as S-PEPS     participant SP as SP      CB-&gt;&gt;S: Handle SP Request     S-&gt;&gt;S: Register Handle SP Request (log)     S-&gt;&gt;SP: Validate Origin     SP--&gt;&gt;S:      </pre>
		<p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>• Citizen-Browser: The S-PEPS receive the Request from a SP through the Citizen-Browser.</li> </ul>
		<p><b>Data</b></p> <pre> graph LR     SPID[SP ID] --&gt; CheckSPAU[Check SP AU Request]     QAALevel[QAA Level] --&gt; CheckSPAU     CheckSPAU --&gt; SPList[(SP's List)]     CheckSPAU --&gt; InitParams[(Init Parameters)]     CheckSPAU --&gt; LogConfig[(Log Configuration)]     CheckSPAU --&gt; SPAuthVal[SP AU Request Validation]     </pre> <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>• SP ID</li> <li>• SP URL</li> <li>• Country of origin (in case a PEPS is shared)</li> <li>• Sector ID</li> <li>• Application name</li> <li>• QAA Level requested by the SP</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>• SP AU Request Validation</li> </ul> <p><b>CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>• SP's list: List of SP's allowed to communicate with this PEPS</li> <li>• Init Parameters</li> </ul>
		<p><b>Error Communications</b></p> <ul style="list-style-type: none"> <li>• Send <u>HTML KO.1</u> to the Citizen-Browser.</li> <li>• If a <u>Handle Error</u> succeed.</li> </ul>

		<b>Actions</b> <ul style="list-style-type: none"> <li>• Get SP AU Request ( )</li> <li>• Register Handle SP AU Request (SP ID, S-PEPS, “SP AU Request”)</li> <li>• Validate Origin (SP ID, QAA Level).</li> </ul> <p>(If SP ID and QAA Level are missing → <a href="#">Handle Error SPAU0101</a>)</p> <ul style="list-style-type: none"> <li>○ If access control is based on SP’s list: <ul style="list-style-type: none"> <li>✓ Check SP (SP ID, SP List): SP ID in SP’s List. (If SP ID is not in the list → <a href="#">Handle Error SPAU0102</a>)</li> <li>✓ Else, check SP Domain (SP Request Domain): validate if the request domain matches the registered SP Domain. (If the origin is not correct → <a href="#">Handle Error SPAU0103</a>)</li> </ul> </li> <li>○ Check number of requests (SP ID): number of requests in the last period of 60 seconds.</li> </ul> <p>(If this number is greater than or equal to the maximum number - to avoid DoS → <a href="#">Handle Error SPAU0104</a>)</p>
2	Generate HTML	<b>Description</b> Generates the HTML that shows the Country Selector form and gets citizen’s nationality selected.
<b>Sequence Diagram</b>  <pre> sequenceDiagram     participant User     participant System     Note over System: 2. Generate HTML     System-&gt;&gt;User: HTML.1   </pre>		
<b>Detailed Sequence Diagram</b>  <pre> sequenceDiagram     participant User     participant System     Note over System: Generate HTML     System-&gt;&gt;User: Present Country Selector     User-&gt;&gt;System:      System-&gt;&gt;System: Register Present Country Selector (log)     System-&gt;&gt;User:    </pre>		
<b>External Actors</b> <ul style="list-style-type: none"> <li>•</li> </ul>		

<p><b>Data</b></p>  <pre> graph LR     SPID[SP ID] --&gt; GenHTML((Generate HTML))     QAALevel[QAA Level] --&gt; GenHTML     GenHTML --&gt; HTMLselector[HTML-country-selector]     SPslist[SP's list] --&gt; GenHTML   </pre> <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>• SP ID</li> <li>• Attributes requested (mandatory/optional)</li> <li>• QAA Level</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>• HTML-country-selector</li> </ul> <p><b>CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>• SP's list (optional)</li> </ul>	<p><b>Error Communications</b></p> <ul style="list-style-type: none"> <li>• None</li> </ul>
<p><b>Actions</b></p> <ul style="list-style-type: none"> <li>• Present Country Selector ( SP ID, QAA Level)</li> </ul>	

*Table 3 – Description sequence for SP without SAML capacities*



### 2.3.1.1.3 Sequence diagram AU

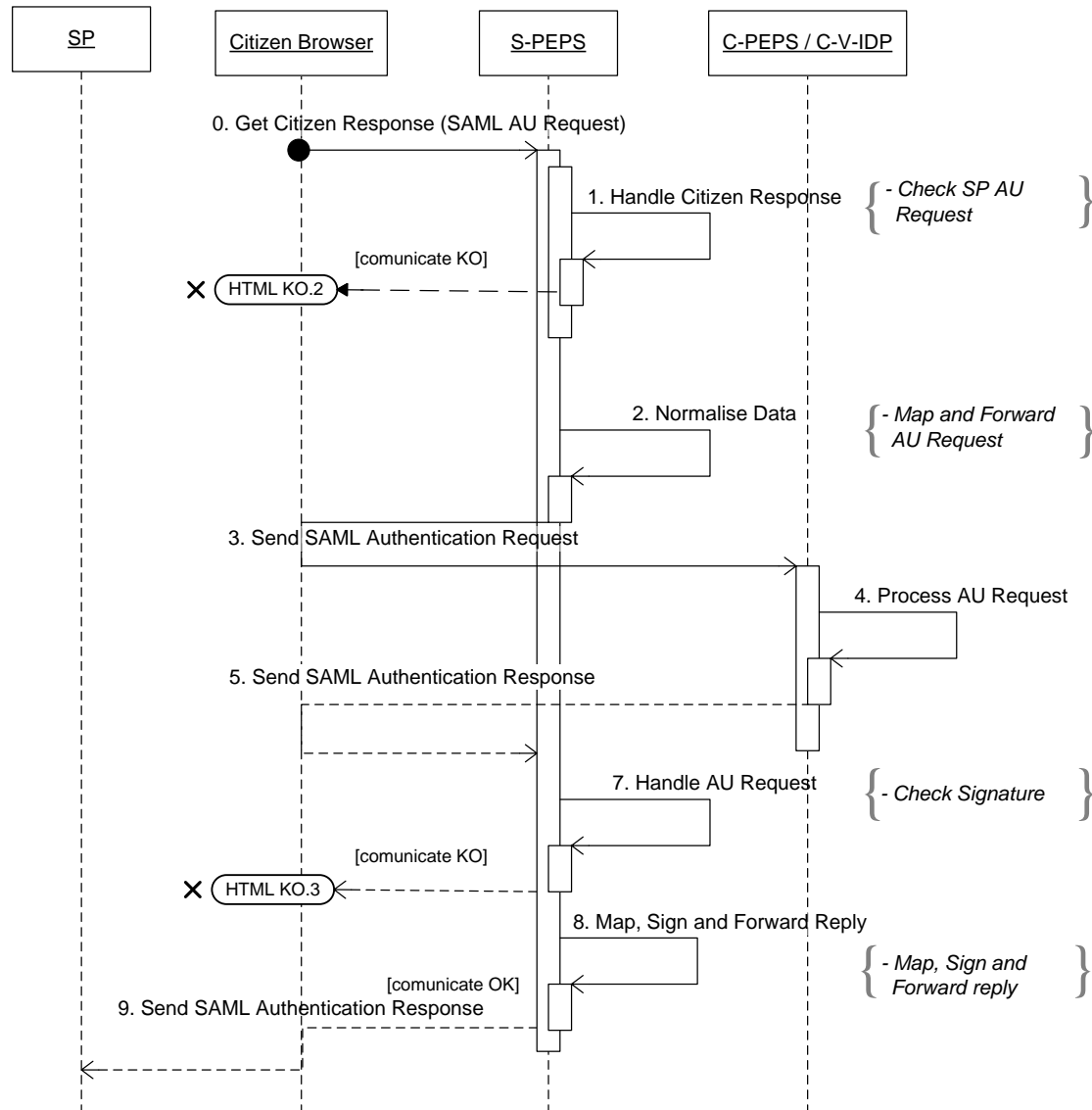
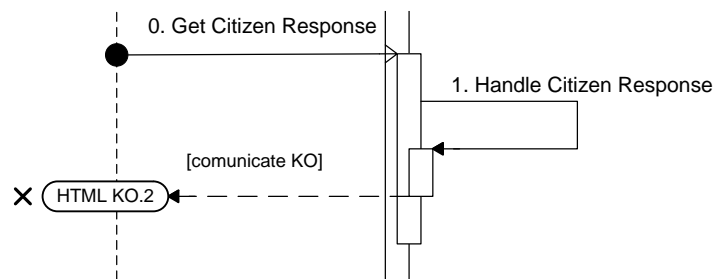


Figure 5: Sequence diagram Authentication in S-PEPS

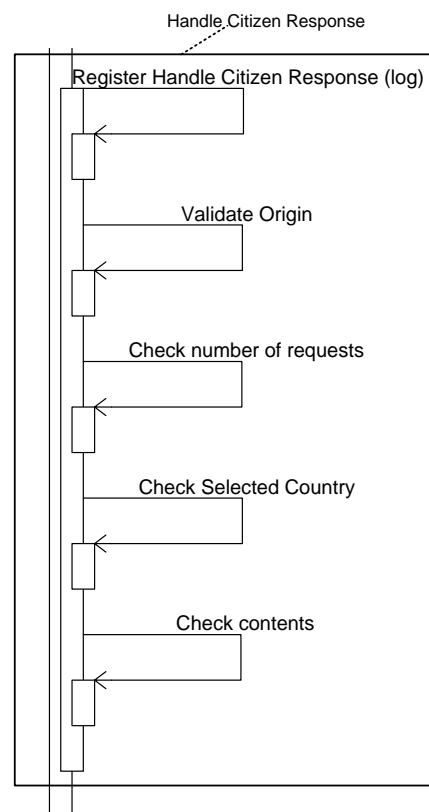
### 2.3.1.1.4 Description

Message sequences (interactions)	Description
0-1 Handle Citizen's Response	<p><b>Description</b></p> <p>Receives Citizen's reply, add it to the AU Request and check AU request validation (this task includes log activity).</p>

### Sequence Diagram



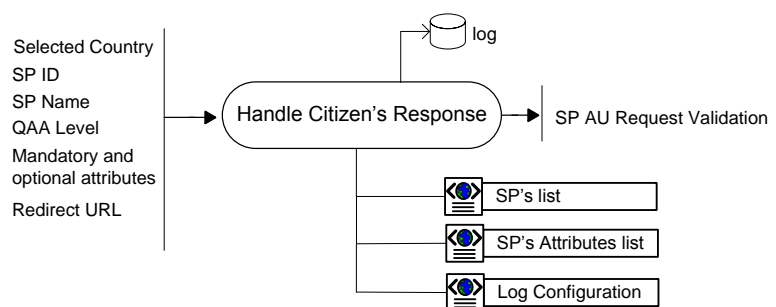
### Detailed Sequence Diagram



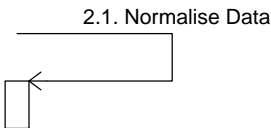
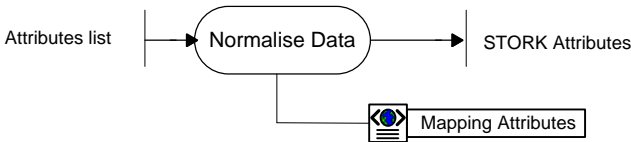
### External Actors

- Citizen-Browser

### Data



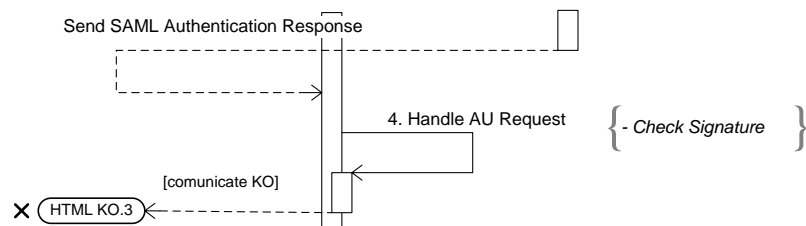
<p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>• Citizen's selected country</li> <li>• SP ID</li> <li>• SP Name</li> <li>• QAA Level</li> <li>• Mandatory and optional Attributes list</li> <li>• Redirect URL</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>• SP AU Request Validation</li> </ul> <p><b>COMMON CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>• SP's list</li> <li>• SP's Attributes List</li> <li>• Log Configuration</li> </ul>	<p><b>Error Communications</b></p> <ul style="list-style-type: none"> <li>• Send <u>HTML KO.2</u> to the Citizen-Browser. If a <a href="#">Handle Error</a> succeed.</li> </ul>
<p><b>Actions</b></p> <ul style="list-style-type: none"> <li>• Get Citizen's Country Selected ( Citizen's Reply)</li> <li>• Validate Origin (SP ID, QAA Level). (If SP ID and QAA Level are missing → <a href="#">Handle Error SPAU0401</a>)</li> </ul> <ul style="list-style-type: none"> <li>○ If Authentication Type is based on SP's list: <ul style="list-style-type: none"> <li>✓ Check SP (SP ID, SP List): SP in SP's List. (If SP is not in the list → <a href="#">Handle Error SPAU0402</a>)</li> <li>✓ Else, check SP Domain (SP Request Domain, Redirect URL): validate if the request domain and Redirect URL matches the registered SP Domain. (If the request domain or Redirection URL is not correct → <a href="#">Handle Error SPAU0403</a>)</li> </ul> </li> <li>○ Check number of requests (SP ID, 60): number of requests in the last period of 60 seconds. (If this number is greater than or equal to the maximum number - to avoid DoS → <a href="#">Handle Error SPAU0404</a>)</li> <li>○ Check Selected Country (SelectedCountry) (If selected country is not a valid country → <a href="#">Handle Error SPAU0405</a>)</li> </ul>	

		<ul style="list-style-type: none"> <li>○ Check contents (Mandatory and optional Attributes list).</li> </ul> <p>(If any Mandatory or optional Attribute isn't in SP's Attributes List → <a href="#">Handle Error SPAU0406</a>)</p> <ul style="list-style-type: none"> <li>○ Register Handle Citizen Response (Citizen, S-PEPS, "Select Country")</li> </ul>
2	Map and Forward AU Request	<p><b>Normalise data and send AU request to colleague PEPS</b></p> <p>This task includes some internal activities to normalise all the received data, log activity and send to Colleague PEPS/V-IDP.</p> <p>2.1 Normalise data</p> <p>2.2 Get SAML Authentication Request</p>
2.1	Normalise data	<p><b>Description</b></p> <p>Normalise all received data, mapping the MS values to STORK nomenclature.</p>
<p><b>Sequence Diagram</b></p>  <pre> sequenceDiagram     participant Actor     Actor Actor-&gt;&gt;2.1. Normalise Data     activate 2.1. Normalise Data     deactivate 2.1. Normalise Data   </pre>		
<p><b>Data</b></p>  <pre> graph LR     A[Attributes list] --&gt; B((Normalise Data))     B --&gt; C[STORK Attributes]     B --&gt; D[Mapping Attributes]   </pre> <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>• Attribute list with values</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>• Attribute names and data values in STORK nomenclature.</li> </ul> <p><b>CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>• Mapping Attributes: Map between STORK names/values and MS names/values for the attributes (text values are not included)</li> </ul>		
2.2	Send SAML Authentication Request	<p><b>Description</b></p> <p>Send AU Request to Colleague PEPS/V-IDP.</p>
<p><b>Sequence Diagram</b></p>		

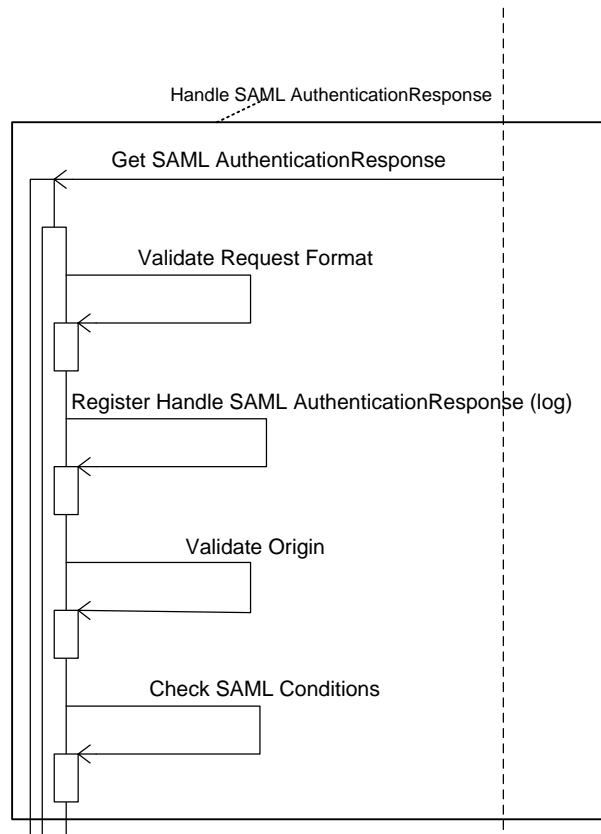
2.2. Send SAML Authentication Request		
<p><b>Detailed Sequence Diagram</b></p>		
<p><b>Data</b></p> <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>• SAML AuthenticationQuery</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>• SAML AuthenticationQuery</li> </ul> <p><b>CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>• Log Configuration</li> </ul>		
<p><b>Actions</b></p> <ul style="list-style-type: none"> <li>• Register Send SAML AuthenticationQuery (S-PEPS, C-PEPS, “Map and Forward AU Request”)</li> </ul>		
3	Process AU Request	<p><b>Description</b></p> <p>The request is processed in the Colleague PEPS/V-IDP. The authentication is performed with the user and the security token created.</p>
<p><b>Sequence Diagram</b></p>		
4	Handle AU Request	<p><b>Description</b></p> <p>S-PEPS receives SAML AuthenticationResponse through the Citizen-Browser issued by Colleague PEPS/V-IDP.</p>

S-PEPS validates SAML AuthenticationResponse signature.

### Sequence Diagram



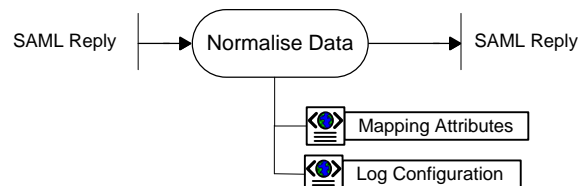
### Detailed Sequence Diagram



### External Actors

- Citizen-Browser: The S-PEPS receives the SAML AU Response from a Colleague C-PEPS/V-IDP through the Citizen-Browser.

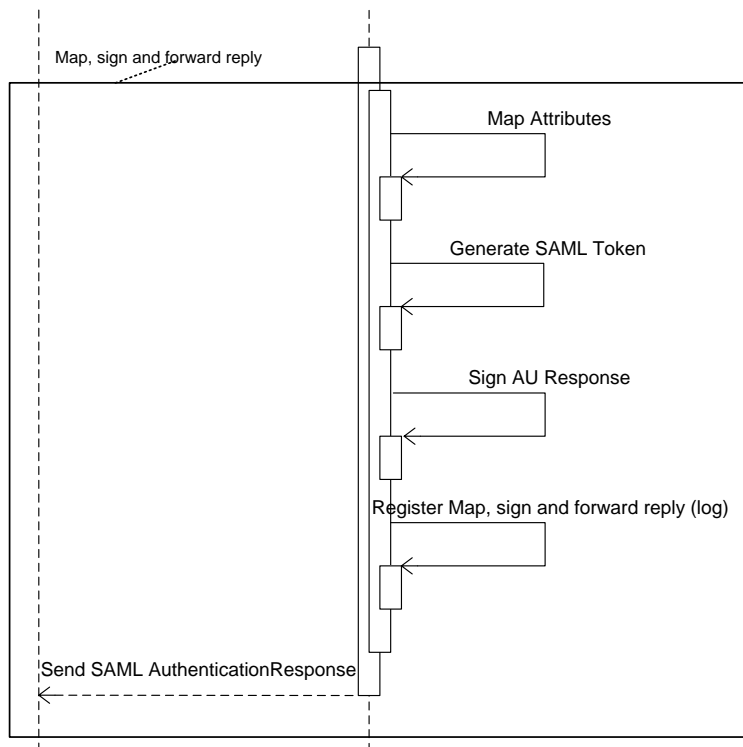
### Data



INPUT

		<ul style="list-style-type: none"> <li>• SAML Reply</li> </ul> <p>OUTPUT</p> <ul style="list-style-type: none"> <li>• SAML Reply</li> </ul> <p>CONFIG FILES NEEDED</p> <ul style="list-style-type: none"> <li>• Mapping attributes</li> <li>• Log Configuration</li> </ul>
		<p><b>Error Communications</b></p> <ul style="list-style-type: none"> <li>• Send <u>HTML KO.3</u> to the Citizen-Browser.</li> </ul> <p>If a <a href="#">Handle Error</a> succeed.</p>
		<p><b>Actions</b></p> <ul style="list-style-type: none"> <li>• Receive SAML AuthenticationResponse ( )</li> <li>• Register Handle SAML AuthenticationResponse (C-PEPS, S-PEPS, “Check Signature”)</li> <li>• Validate Response Format (SAML) (If the format is not correct → <a href="#">Handle Error SPAU2201</a>)</li> <li>• Validate Origin (Colleague PEPS/V-IDP): Validate Colleague PEPS/ V-IDP Signature. (If the origin is not correct → <a href="#">Handle Error SPAU2202</a>)</li> <li>• Check SAML Conditions (notBefore, notAfter, etc.) (If conditions are not fulfilled → <a href="#">Handle Error SPAU2203</a>)</li> </ul>
5	Map, Sign and forward reply	<p><b>Description</b></p> <p>Maps, generates the SAML token, signs and sends it to SP.</p>
		<p><b>Sequence Diagram</b></p> <pre> sequenceDiagram     participant P1     participant P2     participant P3     P1-&gt;&gt;P2: Send SAML Authentication Response     P2--&gt;P1: [communicate OK]     P3-&gt;&gt;P2: 5. Map, Sign and Forward Reply     note over P3: { - Map, Sign and Forward reply }   </pre>

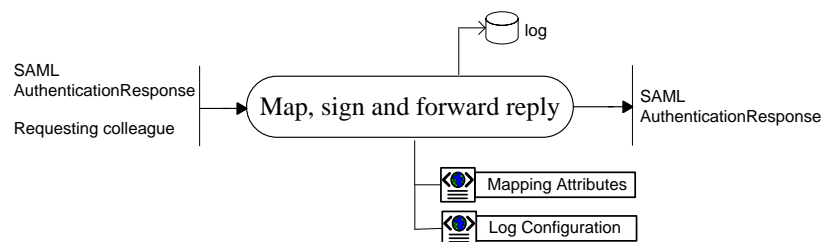
### Detailed Sequence Diagram



### External Actors

- SP

### Data



### INPUT

- SAML AuthenticationResponse
- Requesting colleague PEPS

### OUTPUT

- SAML AuthenticationResponse.

### CONFIG FILES NEEDED

- Mapping Attributes: Map between STORK names/values and MS values for the attributes (text values are not included)
- Log Configuration

### Error Communications

- None.



**Actions**

- Map STORK values to MS values (STORK Values, MS Values)
- Generate SAML Response (Colleague SAML Request)
- Sign SAML Authentication Response (SAML)
- Send SAML Authentication Response ( )
- Register Map, sign and forward reply (S-PEPS, C-PEPS, “Map, sign an forward reply”)

**Table 4 – Description sequence Authentication in S-PEPS****2.3.1.2 Certificate Validation**

The second business process in the S-PEPS is Certificate Validation. This process is designed to support to cross border digital signatures. Although the verification of the signature (file format, mathematical correctness, certificate usage, etc.) can be performed by the SP, for the validation of the status of a certificate the SP needs this cross border verification service.

It is implemented as an Online Certificate Status Protocol (OCSP) service.

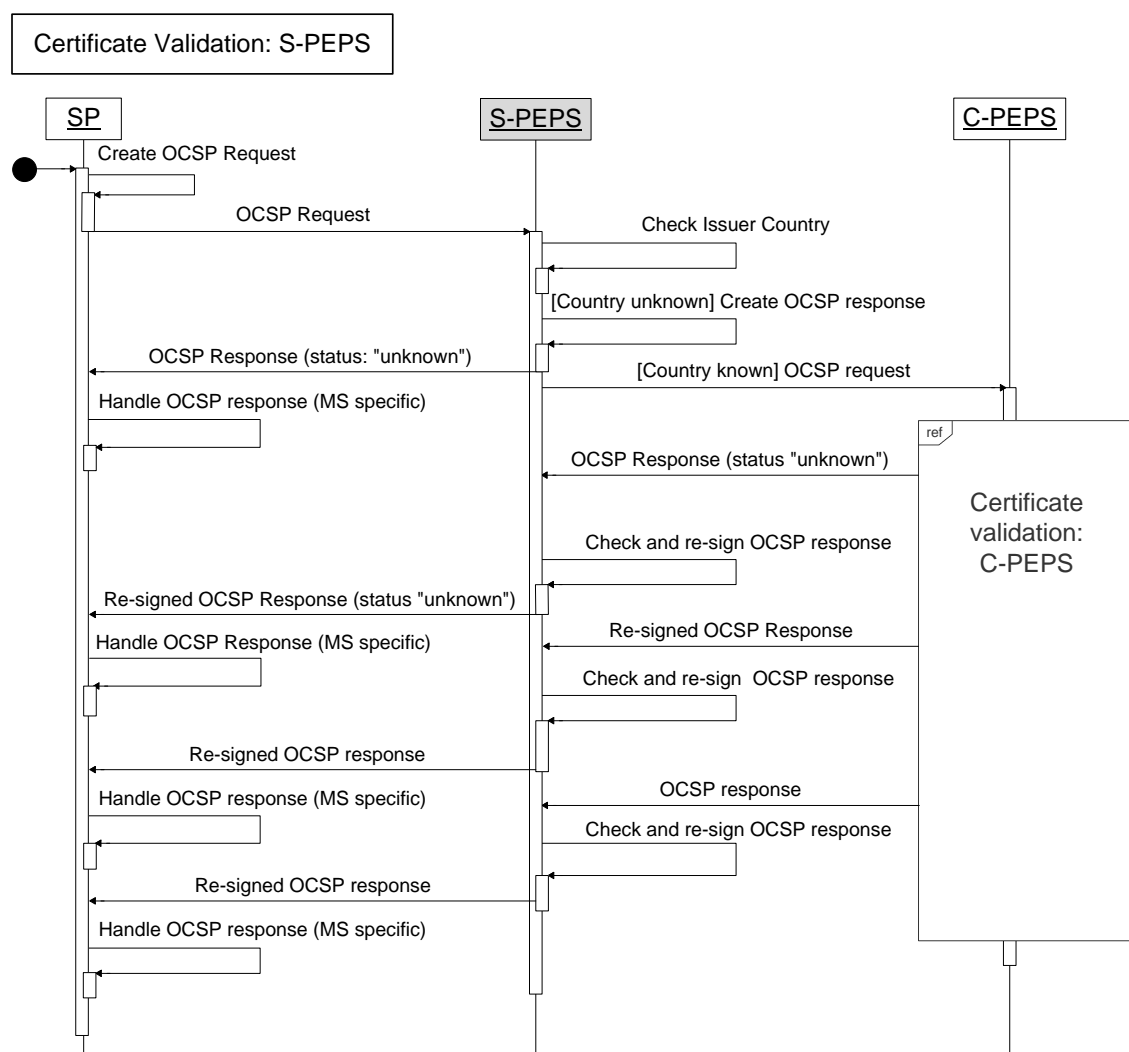
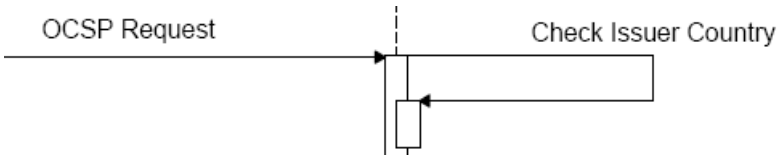
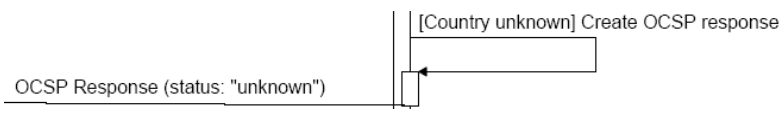
**2.3.1.2.1 Sequence diagram CV**

Figure 6: Sequence diagram Certificate Validation in S-PEPS

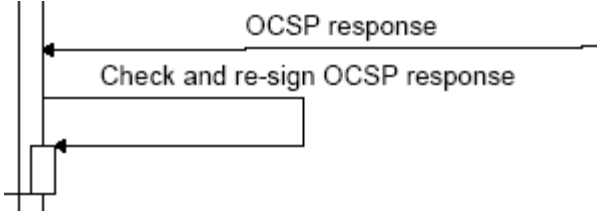
## 2.3.1.2.2 Description

Description		
1	Check Issuer Country	<b>Sequence Diagram</b>  <pre> sequenceDiagram     actor SP     participant Process     SP-&gt;&gt;Process: OCSP Request     activate Process     Process-&gt;&gt;Process: Check Issuer Country     deactivate Process </pre>
		<b>External Actors</b> <ul style="list-style-type: none"> <li>• SP</li> </ul>
		<b>Config files needed</b> <ul style="list-style-type: none"> <li>• colleague_peps_list</li> <li>• SP certificate (necessary in case the OCSP request is signed)</li> </ul>
		<b>Data</b> INPUT <ul style="list-style-type: none"> <li>• OCSP Request</li> </ul> OUTPUT <ul style="list-style-type: none"> <li>• Flag Country known (values: Yes/No)</li> </ul>
		<b>Actions</b> <ul style="list-style-type: none"> <li>• Receive the OCSP Request from the SP</li> <li>• Check OCSP Request</li> <li>• Extract Issuer data (hash algorithm, hash value of the issuer's name, hash of the issuer's public key) from the Cert ID field in the OCSP request</li> <li>• Search in colleague_peps_list the country data (name, location, ..) corresponding to the Issuer data</li> <li>• If a country is found for the Issuer data, the flag Country known takes the value 'yes'; otherwise it takes the value 'No'</li> </ul>
2	[Country unknown] Create OCSP response	<b>Sequence Diagram</b>  <pre> sequenceDiagram     actor SP     participant Process     SP-&gt;&gt;Process: OCSP Response (status: "unknown")     activate Process     Process-&gt;&gt;Process: [Country unknown] Create OCSP response     deactivate Process </pre>
		<b>External Actors</b> <ul style="list-style-type: none"> <li>• SP</li> </ul>
		<b>Config files needed</b> <ul style="list-style-type: none"> <li>• S-PEPS certificate and key</li> </ul>
		<b>Data</b>

		<p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>Flag Country known (Value 'No')</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>OCSP response (status 'unknown')</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li>Construct an OCSP response with the cert status 'unknown'</li> <li>Sign the OCSP response (with the key in S-PEPS certificate and key).</li> <li>Send the OCSP response to the SP.</li> </ul>
3	[Country known] OCSP request	<p><b>Sequence Diagram</b></p> <pre>sequenceDiagram     participant Self     Self-&gt;&gt;Self: [Country known] OCSP request</pre> <p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>Colleague PEPS</li> </ul> <p><b>Config files needed</b></p> <ul style="list-style-type: none"> <li>colleague_peps_list</li> </ul> <p><b>Data</b></p> <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>Flag Country known (Value 'Yes')</li> <li>OCSP request (received at step 1)</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>OCSP request</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li>Extract from the colleague_peps_list the location (e.g. IP address, port number) of PEPS (acting as OCSP Responder)</li> <li>Forward the OCSP request (received at step 1) to colleague PEPS</li> </ul>
4-5	OCSP Response (status "unknown"); Check and re-sign OCSP response	<p><b>Sequence Diagram</b></p> <pre>sequenceDiagram     participant Self     participant External     External-&gt;&gt;Self: OCSP Response (status "unknown")     Self-&gt;&gt;Self: Check and re-sign OCSP response</pre> <p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>Colleague PEPS</li> </ul> <p><b>Config files needed</b></p> <ul style="list-style-type: none"> <li>S-PEPS certificate and key</li> <li>certificate of C-PEPS used for OCSP signing</li> </ul>

5-6 Re-signed OCSF response; Check and re-sign OCSF response

(colleague_peps_list)	
<b>Data</b> INPUT <ul style="list-style-type: none"> <li>• OCSF response with cert status “unknown” (received from colleague PEPS)</li> </ul> OUTPUT <ul style="list-style-type: none"> <li>• OCSF response with cert status “unknown” (re-signed by S-PEPS)</li> </ul>	
<b>Actions</b> <ul style="list-style-type: none"> <li>• Receive from C-PEPS an OCSF response with the status ‘unknown’.</li> <li>• Check the OCSF response.</li> </ul> <p>If an error is encountered, the “responseStatus” in the OCSF response need to be changed to “Internal Error”.</p> <ul style="list-style-type: none"> <li>• Re-sign the OCSF response and send it to the SP.</li> </ul>	
<b>Sequence Diagram</b> <pre> sequenceDiagram     participant Process     participant Actor     Note over Process: Check and re-sign OCSF response     Process-&gt;&gt;Process:      Process-&gt;&gt;Actor: Re-signed OCSF Response   </pre>	
<b>External Actors</b> <ul style="list-style-type: none"> <li>• Colleague PEPS</li> <li>• SP</li> </ul>	
<b>Config files needed</b> <ul style="list-style-type: none"> <li>• S-PEPS certificate and key</li> <li>• certificate of C-PEPS used for OCSF signing (colleague_peps_list)</li> </ul>	
<b>Data</b> INPUT <ul style="list-style-type: none"> <li>• OCSF response, which has been re-signed by the colleague PEPS</li> </ul> OUTPUT <ul style="list-style-type: none"> <li>• OCSF response re-signed by S-PEPS</li> </ul>	
<b>Actions</b> <ul style="list-style-type: none"> <li>• Receive the OCSF response from C-PEPS.</li> <li>• Check the OCSF response.</li> </ul> <p>If an error is encountered, the “responseStatus” in the OCSF</p>	

7	OCSP response; Check and re-sign OCSP response	<p>response need to be changed to “Internal Error”.</p> <ul style="list-style-type: none"> <li>• Re-sign the OCSP response and send it to the SP.</li> </ul> <p><b>Sequence Diagram</b></p>  <p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>• Colleague PEPS</li> <li>• SP</li> </ul> <p><b>Config files needed</b></p> <ul style="list-style-type: none"> <li>• S-PEPS certificate and key</li> <li>• certificate of C-PEPS used for OCSP signing (colleague_peps_list)</li> </ul> <p><b>Data</b></p> <p>INPUT</p> <ul style="list-style-type: none"> <li>• OCSP response, which has been issued by the colleague PEPS</li> </ul> <p>OUTPUT</p> <ul style="list-style-type: none"> <li>• OCSP response signed by S-PEPS</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li>• Receive the OCSP response from C-PEPS.</li> <li>• Check the OCSP response.</li> </ul> <p>If an error is encountered, the “responseStatus” in the OCSP response need to be changed to “Internal Error”.</p> <ul style="list-style-type: none"> <li>• Re-sign the OCSP response and send it to the SP.</li> </ul>
---	--	--

*Table 5 – Description sequence Certificate Verification in S-PEPS*

### 2.3.2 C-PEPS

The C-PEPS is the PEPS in the role of verification of citizen’s credentials. This role is also composed of two business processes:

1. Authentication
2. Certificate validation (to support digital signature)

## 2.3.2.1 Authentication process

### 2.3.2.1.1 Sequence diagram AU

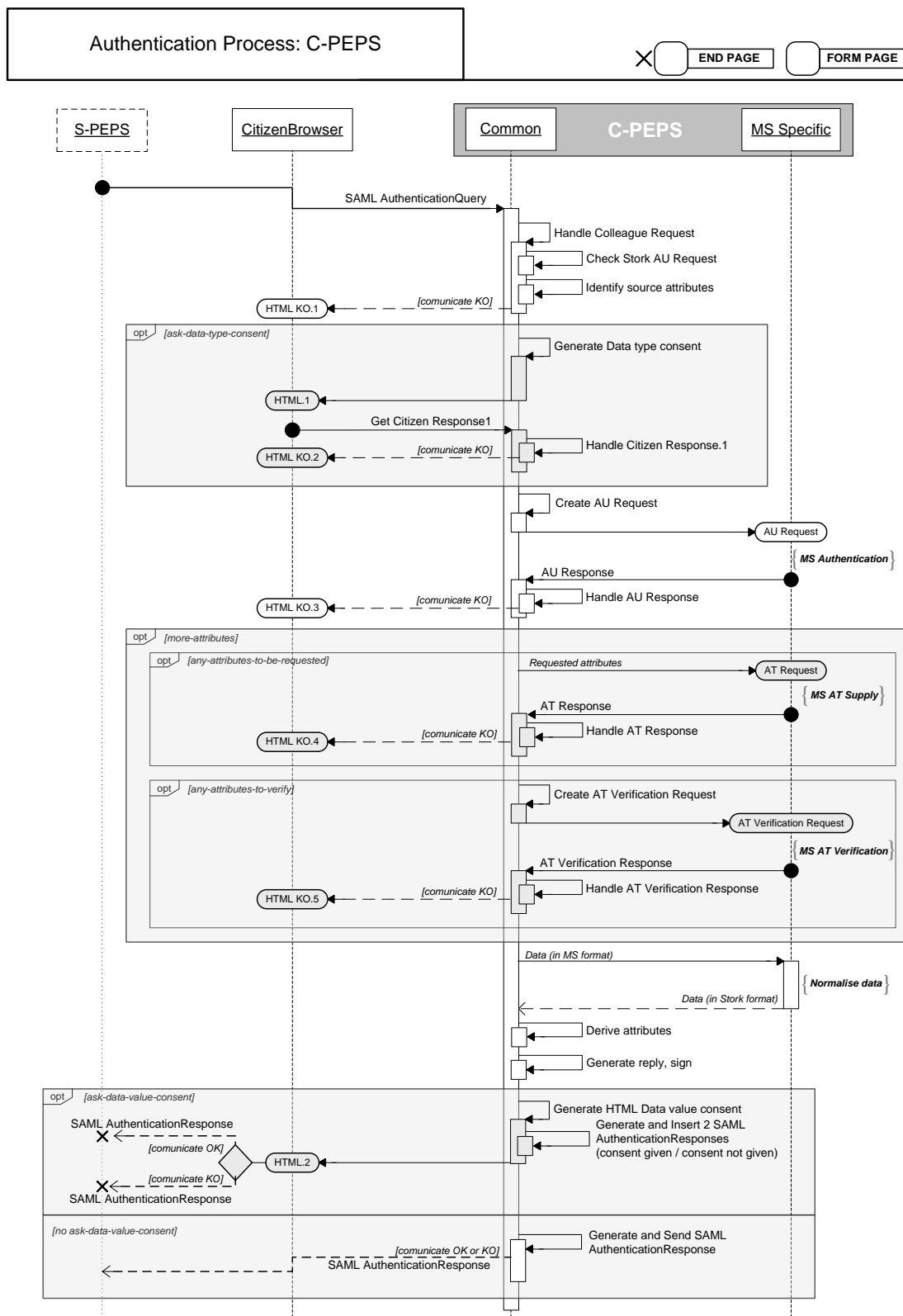
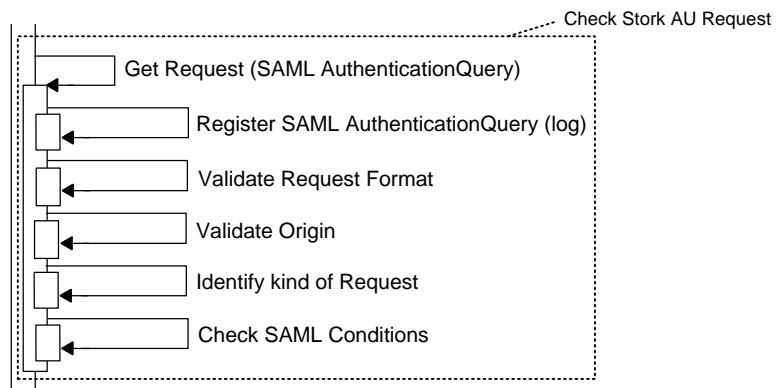
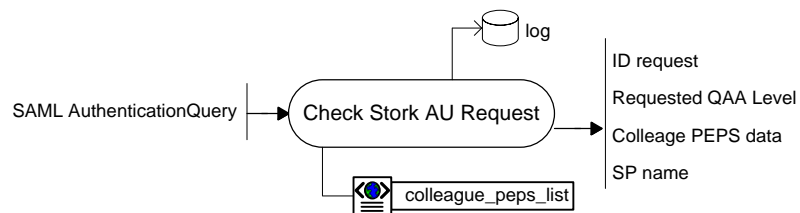


Figure 7: Sequence diagram Authentication in C-PEPS

## 2.3.2.1.2 Description

Message sequences (interactions)	
1 Handle Colleague Request	<b>Description</b> Handles request of colleague PEPS or V-IDP This task includes some internal activities to validate the request received and prepare the next steps towards the user authentication. 1.1 Check Authentication Request 1.2 Identify source attributes
	<b>External Actors</b> <ul style="list-style-type: none"> <li>Citizen-Browser: The C-PEPS receive the Request form a colleague PEPS or V-IDP through the Citizen-Browser.</li> </ul>
	<b>Sequence Diagram</b> <pre> sequenceDiagram     participant Start     participant Actor     participant Process     Start-&gt;&gt;Actor: SAML AuthenticationQuery     activate Actor     Actor-&gt;&gt;Process: Handle Colleague Request     activate Process     Process-&gt;&gt;Actor: HTML KO.1     deactivate Process     Actor-&gt;&gt;End: [communicate KO]     deactivate Actor           </pre>
	<b>Error Communications</b> <ul style="list-style-type: none"> <li>Send <u>HTML KO.1</u> to the Citizen-Browser. (If an error succeeds a <a href="#">Handle Error</a> function manages this error).</li> </ul>
1.1 Check authentication request	<b>Description</b> The objective of this activity is to register the request, check the origin of the request and decide if the request, in this first step, is accepted or not.
	<b>Sequence Diagram</b> <pre> sequenceDiagram     participant Start     participant Actor     participant Process     Start-&gt;&gt;Actor:      activate Actor     Actor-&gt;&gt;Process: Check Stork AU Request     activate Process     Process-&gt;&gt;Actor:      deactivate Process     Actor-&gt;&gt;End:      deactivate Actor           </pre>

**Detailed Sequence Diagram****Data****INPUT**

- SAML AuthenticationQuery

**OUTPUT**

- ID request
- Requested QAA Level
- Colleague PEPS data
- SP name

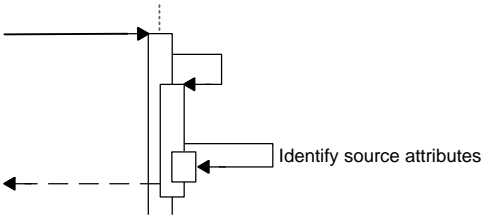
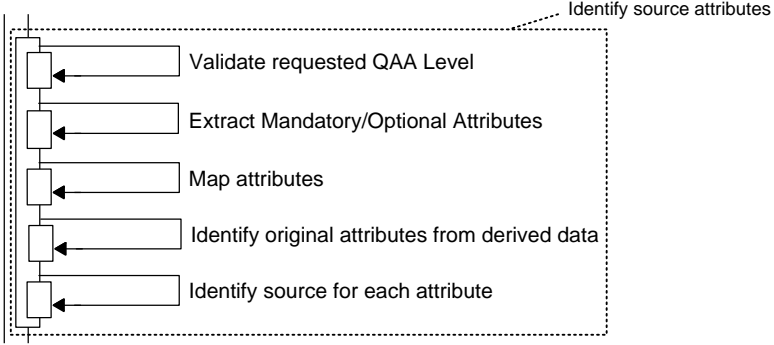
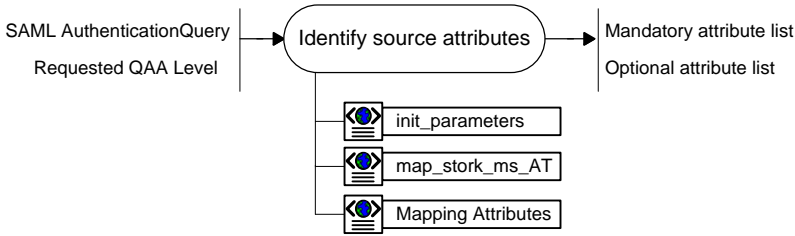
**CONFIG FILES NEEDED**

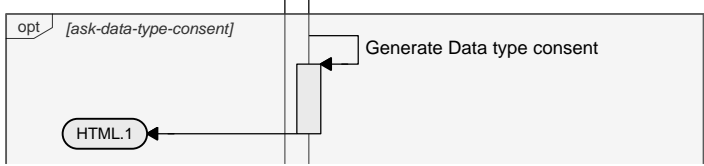
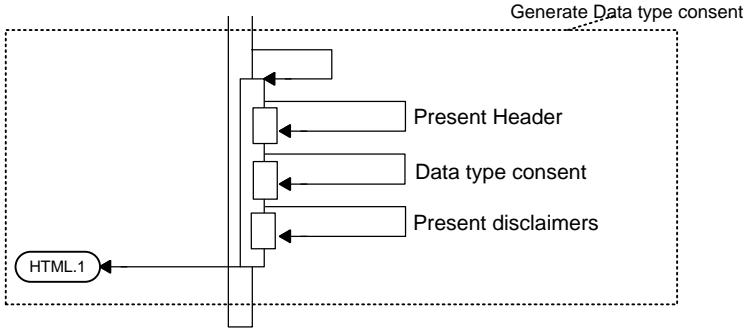
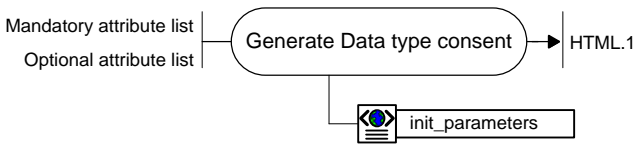
- colleague\_peps\_list: List of colleagues that form STORK

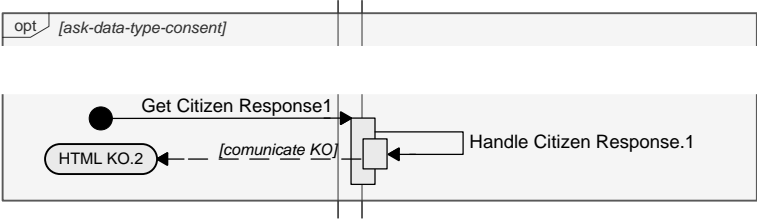
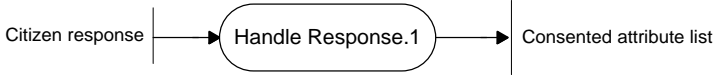
**Actions**

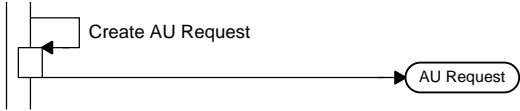

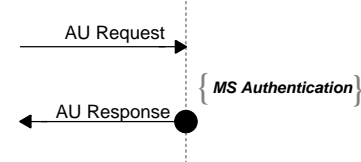
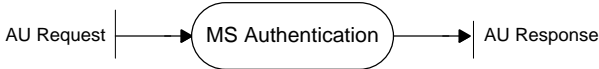
- Get Request (SAML AuthenticationQuery): Captures any Request from Citizens browser send to the C-PEPS.
- Register Request (SAML AuthenticationQuery): Log the SAML request received.
- Validate Request Format (SAML)  
(If the format is not correct → [Handle Error CPAU0101](#))
- Validate Origin (Colleague PEPS): Validate Colleague PEPS Signature. This data is available in the colleague peps list file.  
(If the origin is not correct → [Handle Error CPAU0102](#))
- Identify kind of Request: Extract Request and identify kind of Request (AuthenticationQuery, AttributeQuery, etc). In this activity we analyse the AuthenticationQuery case.
- Check SAML Conditions (notBefore, notAfter, etc.)  
(If conditions are not met → [Handle Error CPAU0103](#))

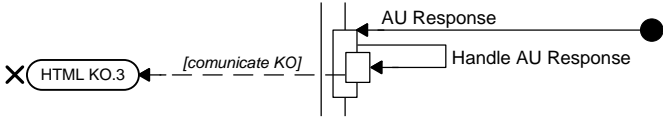
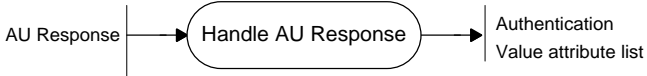


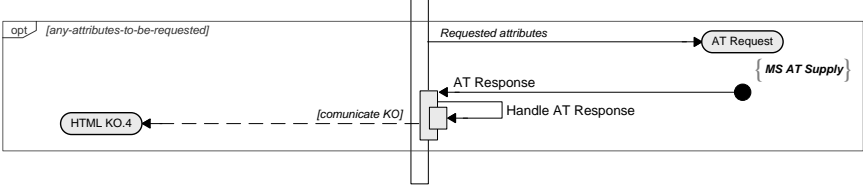
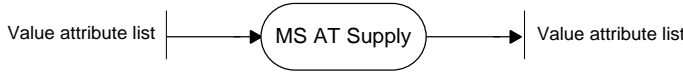
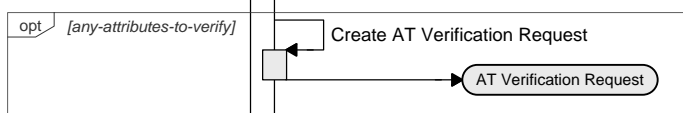

1.2 Identify source attributes	<div data-bbox="531 194 1417 286"> <p><b>Description</b></p> <p>This activity checks the validity of the request in terms of contents.</p> </div> <div data-bbox="531 286 1417 600"> <p><b>Sequence Diagram</b></p>  </div> <div data-bbox="531 600 1417 1055"> <p><b>Detailed Sequence Diagram</b></p>  </div> <div data-bbox="531 1055 1417 1805"> <p><b>Data</b></p>  <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>• SAML AuthenticationQuery</li> <li>• Requested QAA Level</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>• Mandatory requested attributes in MS Attribute nomenclature</li> <li>• Optional attributes in MS Attribute nomenclature</li> </ul> <p><b>CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>• init_parameters (initial C-PEPS parameters)</li> <li>• map_stork_ms_attributes (STORK attributes mapped to MS attributes)</li> </ul> </div> <div data-bbox="531 1805 1417 2018"> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li>• <u>Validate requested QAA Level</u>: Compare requested QAA Level with the CPEPS max QAA Level. (If max level is lower than requested → <a href="#">Handle Error CPAU0104</a>)</li> <li>• <u>Extract mandatory/optional attributes</u>: Identify the mandatory attributes.</li> </ul> </div>
--------------------------------	---

	<p><b>Map attributes:</b> Transform and Derive Stork requested attributes in MS requested attributes.</p> <ul style="list-style-type: none"> <li>• <u>Identify original attributes from derived data:</u> This attributes will be added to the attributes to be requested to the national IDPs, APS. This attributes may be also mandatory or optional. They inherit this characteristic from the derived data.</li> <li>• <u>Identify source for each attribute:</u> For each MS attribute obtained, the national source that can disclosure this data is identified: <ul style="list-style-type: none"> <li>○ available</li> <li>○ not available</li> </ul> <p>(If there is “not source available” for a mandatory attribute → <a href="#">Handle Error CPAU0105</a>)</p> </li> </ul>
<p>2      Generate Data type consent (optional)</p>	<p><b>Description</b></p> <p>This task includes some internal activities to generate the page in which the user is requested to give his consent to the transfer of his data (types).</p> <p>It's optional, only executes if the ask-data-type-consent is set to YES in the <i>init_parameters</i> config file.</p> <p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>• Citizen-Browser: Receives a generated HTML</li> </ul> <p><b>Sequence Diagram</b></p>  <pre> sequenceDiagram     participant User     participant System     opt [ask-data-type-consent]         System-&gt;&gt;System: Generate Data type consent         System--&gt;&gt;User: HTML.1     end   </pre> <p><b>Detailed Sequence Diagram</b></p>  <pre> sequenceDiagram     participant User     participant System     participant System as Generate Data type consent     System-&gt;&gt;System: Present Header     System-&gt;&gt;System: Data type consent     System-&gt;&gt;System: Present disclaimers     System--&gt;&gt;User: HTML.1   </pre> <p><b>Data</b></p>  <pre> graph LR     Input["Mandatory attribute list Optional attribute list"] --&gt; Process((Generate Data type consent))     InitParameters[init_parameters] --&gt; Process     Process --&gt; Output[HTML.1]   </pre> <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>• Requested attributes (mandatory / optional) in C-PEPS's native language</li> </ul> <p><b>OUTPUT</b></p>

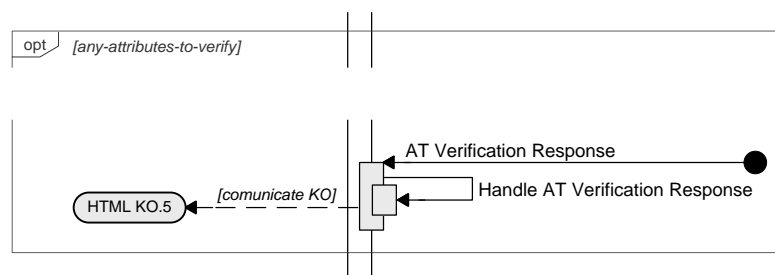
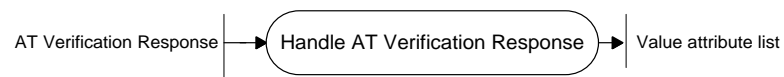
	<ul style="list-style-type: none"> <li>HTML.1</li> </ul> <p>CONFIG FILES NEEDED</p> <ul style="list-style-type: none"> <li>init_parameters</li> </ul> <p><b>Error Communications</b></p> <ul style="list-style-type: none"> <li>None</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li><u>Evaluate ask-data-type-consent</u> from init parameters file.</li> <li><u>Present header</u> (SP name)</li> <li><u>Data type consent</u>: Present attribute list (requested attributes: mandatory and optional) and gets citizen's consent. Mandatory attributes cannot be deselected.</li> <li><u>Present disclaimers</u></li> </ul>
3	<p>Handle Citizen Response.1 (optional)</p> <p><b>Description</b> Receives citizen's reply.</p> <p><b>Sequence Diagram</b></p>  <pre> sequenceDiagram     participant Actor     Actor-&gt;&gt;Actor: Get Citizen Response1     Actor-&gt;&gt;Actor: Handle Citizen Response.1     Actor--&gt;&gt;Actor: HTML KO.2     Note over Actor: [ask-data-type-consent]     Actor--&gt;&gt;Actor: [communicate KO]   </pre> <p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>Citizen</li> </ul> <p><b>Data</b></p>  <pre> graph LR     Input[Citizen response] --&gt; Process([Handle Response.1])     Process --&gt; Output[Consented attribute list]   </pre> <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>Citizen reply</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>Consented attribute list (mandatory and optional)</li> </ul> <p>CONFIG FILES NEEDED</p> <ul style="list-style-type: none"> <li>None</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li><u>Register citizen reply</u> (log citizen's consent)</li> <li><u>Check citizen reply format</u> (If the response is malformed → <a href="#">Handle Error CPAU0301</a>)</li> <li><u>Check consents</u> (If the consent is not given for a mandatory attribute → <a href="#">Handle Error CPAU0302</a>)</li> </ul>

	<b>Error Communications</b> <ul style="list-style-type: none"> <li>Send <u>HTML KO.2</u> to the Citizen-Browser. (If a <u>Handle Error</u> succeed).</li> </ul>
4 Create AU Request	<b>Description</b> Create an AU Request and send it to the “MS Authentication”.  <b>Sequence Diagram</b>   <b>Data</b>   <b>INPUT</b> <ul style="list-style-type: none"> <li>Consented attribute list (mandatory and optional)</li> <li>Requested QQA Level</li> </ul> <b>OUTPUT</b> <ul style="list-style-type: none"> <li>Authentication Request</li> </ul> <b>CONFIG FILES NEEDED</b> <ul style="list-style-type: none"> <li>init_parameters: (localization of the MS Authentication Module)</li> </ul> <b>Actions</b> <ul style="list-style-type: none"> <li><u>Create Authentication Request</u></li> <li><u>Send Authentication Request to MS Authentication Module</u></li> </ul>
5 MS Authentication	<b>Description</b> Select and Perform Authentication in the IDP/CA. Receive an Authentication Request and return an Authentication Response.  <b>Sequence Diagram</b>   <b>Data</b>   <b>INPUT</b> <ul style="list-style-type: none"> <li>AU Request               <ul style="list-style-type: none"> <li>Consented Attribute list</li> </ul> </li> </ul>

	<ul style="list-style-type: none"> <li>○ Requested QQA Level</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>• AU Response             <ul style="list-style-type: none"> <li>○ Authentication (yes/no)</li> <li>○ Value Attribute list</li> </ul> </li> </ul> <p><b>CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>• None</li> </ul> <p><b>Actions</b></p> <p>Is MS specific.</p> <p>Authentication is a country specific activity or group of activities. Within this activity, some of the requested attributes may be collected.</p>
6      Handle AU Response	<p><b>Description</b></p> <p>Handles an authentication response.</p> <p><b>Sequence Diagram</b></p>  <p><b>Data</b></p>  <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>• AU Response</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>• Authentication (yes/no)</li> <li>• Value attribute list             <ul style="list-style-type: none"> <li>○ Filled attribute list: attributes with found values</li> <li>○ Empty attribute list: attributes without values</li> </ul> </li> </ul> <p><b>CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>• None</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li>• <u>Check AU Request</u> (If the authentication not success → <a href="#">Handle Error CPAU0601</a>)</li> <li>• <u>More attributes?:</u> Evaluate if all attributes needed are available or if more attributes should be requested to MS Attribute Supply. (Also evaluate if some attributes has to be introduced by the user and verified)</li> </ul> <p><b>Error Communications</b></p> <ul style="list-style-type: none"> <li>• Send <u>HTML KO.3</u> to the Citizen-Browser.</li> </ul>

	(If a <a href="#">Handle Error</a> succeed).
7 MS Attribute Supply (optional)	<p><b>Description</b></p> <p>Find attribute values from Attribute providers.</p> <p><b>Sequence Diagram</b></p>  <p><b>Data</b></p>  <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>Value attribute list</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>Value attribute list</li> </ul> <p><b>CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>None</li> </ul> <p><b>Actions</b></p> <p>Is MS specific</p> <p>If values for a mandatory attribute not found → <a href="#">Handle Error CPAU0701</a></p> <p><b>Error Communications</b></p> <ul style="list-style-type: none"> <li>Send <a href="#">HTML KO.4</a> to the Citizen-Browser.</li> </ul> <p>(If a <a href="#">Handle Error</a> succeed).</p>
8 Create AT Verification Request (optional)	<p><b>Description</b></p> <p>Create an Attribute Verification Request and send it to the “MS AT Verification”.</p> <p><b>Sequence Diagram</b></p>  <p><b>Data</b></p>  <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>Attribute list to verify (mandatory and optional)</li> <li>Requested QQA Level</li> </ul>

		<p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>Attribute Verification Request</li> </ul> <p><b>CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>None</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li><u>Create Attribute Verification Request</u></li> </ul>
9	MS AT Verification (optional)	<p><b>Description</b></p> <p>Presents an html to the citizen where he can introduce the attributes that need to be verified against national APs.</p> <p>Verifies attributes given by the citizen checking them at the Attribute provider.</p> <p><b>Sequence Diagram</b></p> <pre> sequenceDiagram     participant User     participant Process     User-&gt;&gt;Process: AT Verification Request     Process--&gt;&gt;User: AT Verification Response     Note right of Process: { MS AT Verification }   </pre> <p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>Citizen</li> </ul> <p><b>Data</b></p> <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>Attribute Verification Request <ul style="list-style-type: none"> <li>Attribute list to verify</li> <li>Requested QQA Level</li> </ul> </li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>Attribute Verification Response <ul style="list-style-type: none"> <li>Value Attribute list</li> </ul> </li> </ul> <p><b>CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>None</li> </ul> <p><b>Actions</b></p> <p>Is MS specific.</p> <p>This activity interacts with the citizen.</p>
10	Handle Attribute Verification Response (optional)	<p><b>Description</b></p> <p>Handles an attribute verification response.</p>

**Sequence Diagram****Data****INPUT**

- AT Verification Response

**OUTPUT**

- Value attribute list

**CONFIG FILES NEEDED**

- None

**Actions**

- Check AT Verification Request  
(If the verification of a mandatory attribute fails → [Handle Error CPAU0801](#))

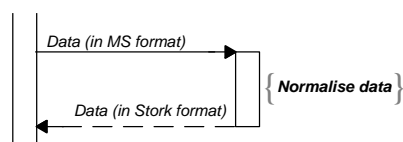
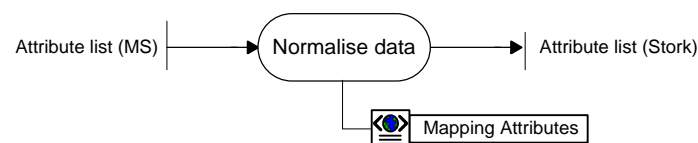
**Error Communications**

- Send HTML KO.5 to the Citizen-Browser.  
(If a [Handle Error](#) succeed).

11 Normalise data

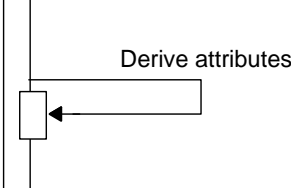

**Description**

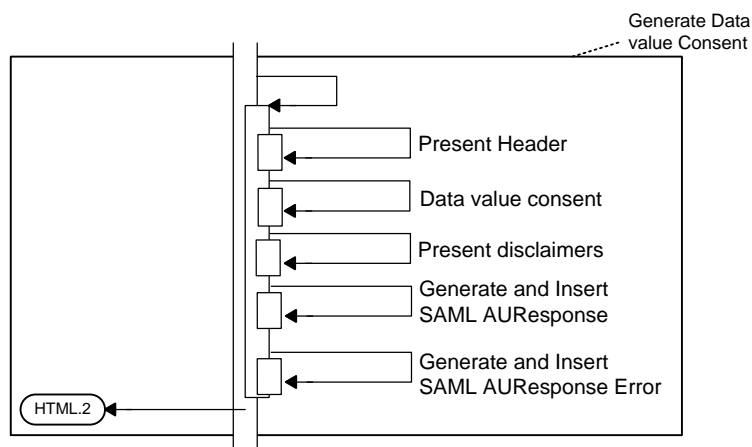
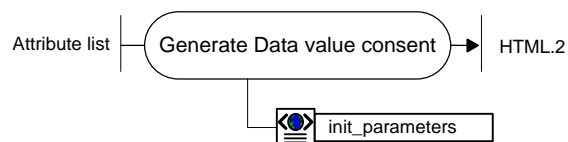
Normalises all received data (from MS Authentication, MS AT Supply and MS AT Verification) mapping the MS names/values to STORK nomenclature.

**Sequence Diagram****Data****INPUT**

- Attribute list (names and data values in MS nomenclature)



	<p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>Attribute list (names and data values in STORK nomenclature)</li> </ul> <p><b>CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>None</li> </ul> <p><b>Actions</b></p> <p>Is MS specific. (text values are not mapped)</p>
<p>12      Derive attributes</p>	<p><b>Description</b></p> <p>Derived attribute data is constructed</p> <p><b>Sequence Diagram</b></p>  <pre> sequenceDiagram     participant Self     Self-&gt;&gt;Self: Derive attributes   </pre> <p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>None</li> </ul> <p><b>Data</b></p> <p><b>INPUT</b></p> <ul style="list-style-type: none"> <li>Attributes and values: Date of Birth, Age threshold (s), eIdentifier</li> </ul> <p><b>OUTPUT</b></p> <ul style="list-style-type: none"> <li>Derived data: Age, IsAgeOver &lt;threshold&gt;, eIdentifier</li> </ul> <p><b>CONFIG FILES NEEDED</b></p> <ul style="list-style-type: none"> <li>none</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li>Derive age</li> <li>Compare with threshold (age)</li> <li>Creates the eIdentifier</li> </ul>
<p>13      Generate HTML Data value consent (optional)</p>	<p><b>Description</b></p> <p>This task includes some internal activities to generate the page in which the user is requested to give his consent to the transfer of his data types and values in the C-PEPS native language except for derive data.</p> <p>It's optional, only executes if the ask-data-value-consent is set to YES in the <i>init_parameters</i> config file.</p> <p>If the Citizen gives data-value-consent, a SAML Authentication Response is sent to the SP, if the data-value-consent is not given another SAML Authentication Response (Error) is sent.</p> <p><b>Sequence Diagram</b></p>  <pre> sequenceDiagram     opt [ask-data-value-consent]         GenerateHTML[Generate HTML Data value consent]         GenerateSAML[Generate and Insert 2 SAML AuthenticationResponses (consent given / consent not given)]         GenerateHTML-&gt;&gt;GenerateSAML         GenerateSAML-&gt;&gt;Decision{ }         Decision-&gt;&gt;SAML_OK[SAML AuthenticationResponse] : [communicate OK]         Decision-&gt;&gt;SAML_KO[SAML AuthenticationResponse] : [communicate KO]     end   </pre>

**Detailed Sequence Diagram****Data****INPUT**

- Attributes list in MS nomenclature

**OUTPUT**

- HTML.2

**CONFIG FILES NEEDED**

- init\_parameters

**Actions**

- Present destination (SP name)
- Present attribute list (requested attributes) with values
- Present disclaimers

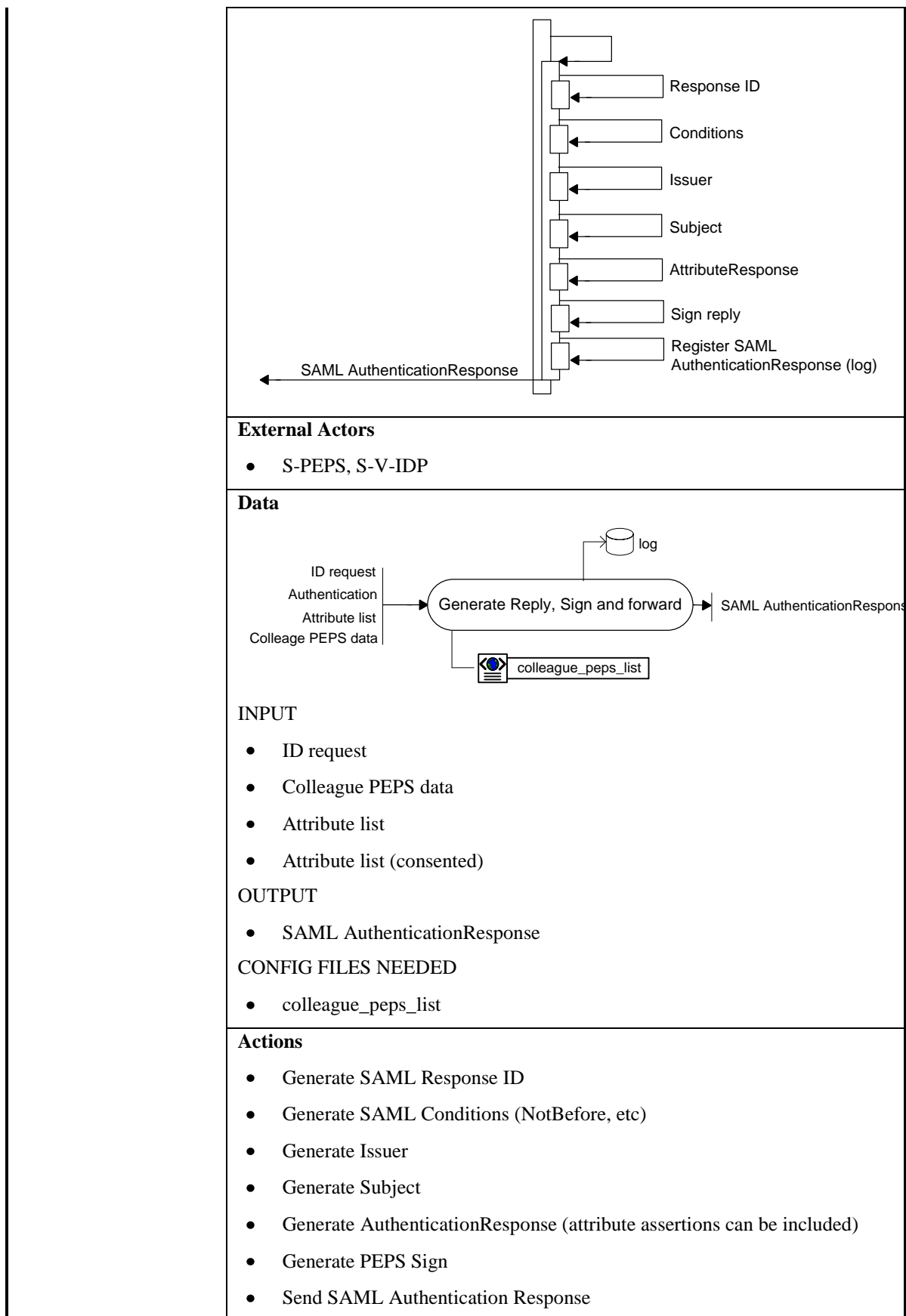
14 Generate and Send SAML Authentication Response

**Description**

Generates the SAML token, and signs it.

It's optional, only executes if the ask-data-value-consent is set to NO in the *init\_parameters* config file.

**Sequence Diagram****Detailed Sequence Diagram**



*Table 6 –Description sequence Authentication in C-PEPS*

### 2.3.2.2 Certificate Validation

Certificate Validation is the other business process in the C-PEPS. As explained in the CV business process in the S-PEPS, it's meant to complement the digital signatures functionality in service providers.

## 2.3.2.2.1 Sequence diagram CV

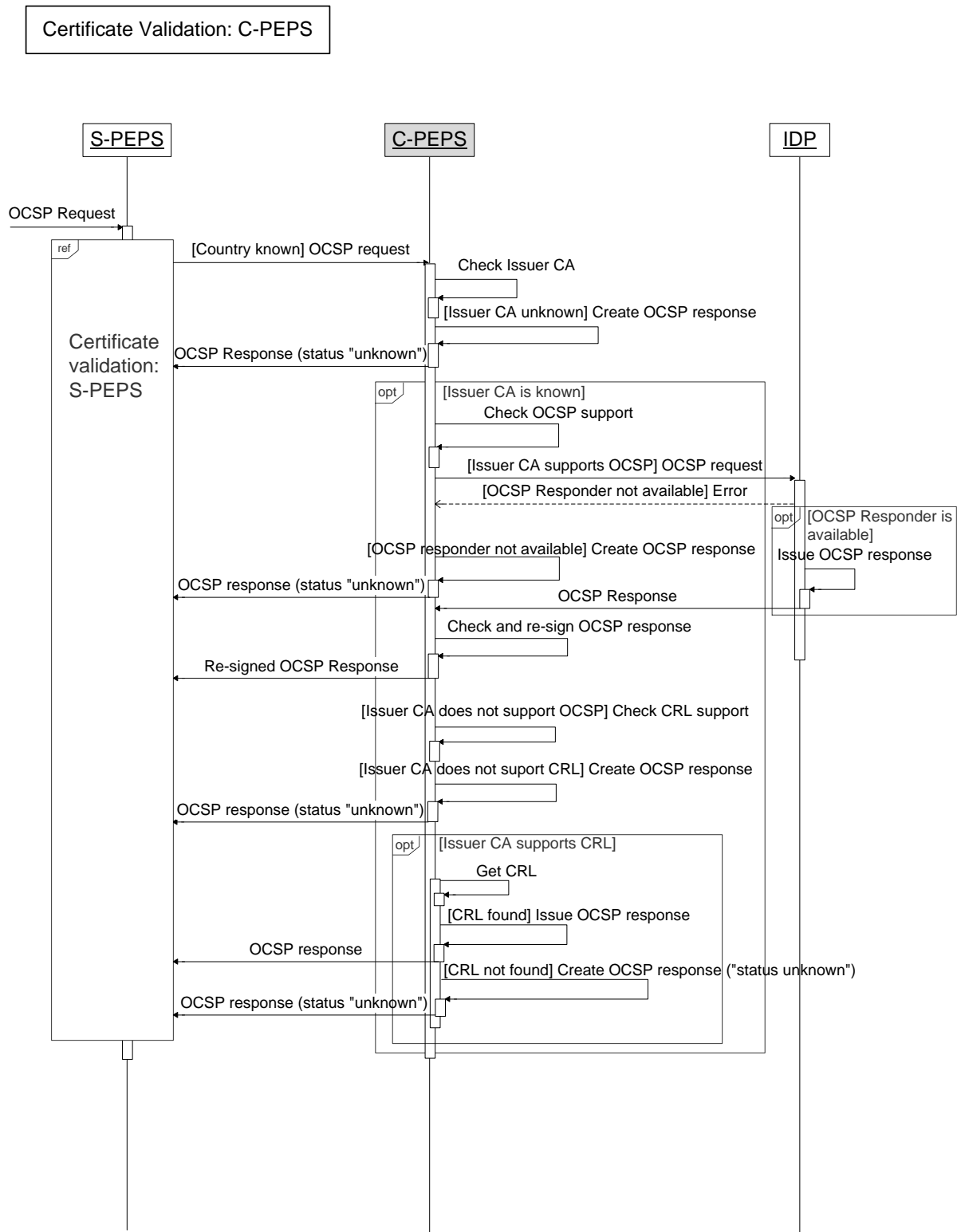


Figure 8: Sequence diagram Certificate Validation in C-PEPS

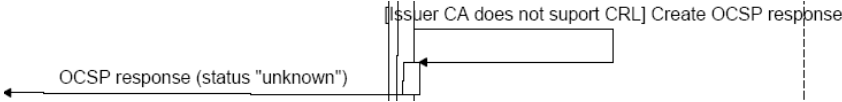
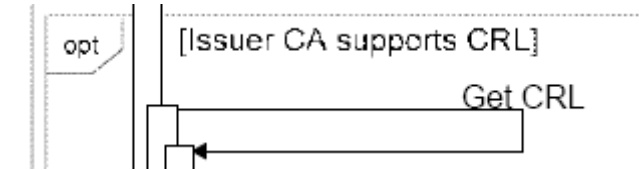
## 2.3.2.2.2 Description

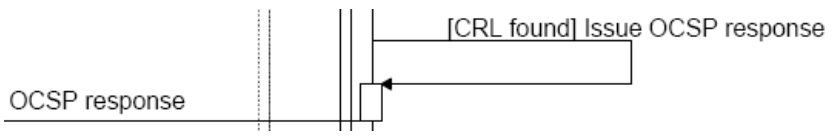
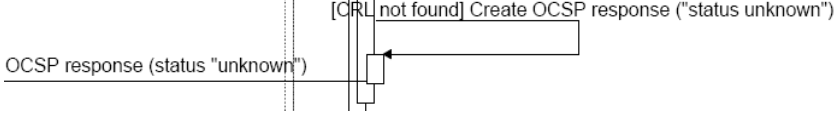
Description		
1	Check Issuer CA	<p><b>Sequence Diagram</b></p> <pre> sequenceDiagram     participant P     participant S     P-&gt;&gt;S: [Country known] OCSF request     S-&gt;&gt;S: Check Issuer CA   </pre> <p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>• Colleague PEPS</li> </ul> <p><b>Config files needed</b></p> <ul style="list-style-type: none"> <li>• Issuer CA list</li> <li>• SP_list (only if the OCSF request is signed)</li> </ul> <p><b>Data</b></p> <p>INPUT</p> <ul style="list-style-type: none"> <li>• OCSF request</li> </ul> <p>OUTPUT</p> <ul style="list-style-type: none"> <li>• Flag Issuer CA known (Yes/No)</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li>• Get OCSF request</li> <li>• Check OCSF request (if the OCSF request is signed then the SP_list is required)</li> <li>• Check whether the issuer CA of the certificate is known (Issuer CA list)</li> </ul>
2	[Issuer CA unknown] Create OCSF response	<p><b>Sequence Diagram</b></p> <pre> sequenceDiagram     participant S     participant P     S-&gt;&gt;S: [Issuer CA unknown] Create OCSF response     S-&gt;&gt;P: OCSF Response (status "unknown")   </pre> <p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>• Colleague PEPS</li> </ul> <p><b>Config files needed</b></p> <ul style="list-style-type: none"> <li>• C-PEPS certificate and key</li> </ul> <p><b>Data</b></p> <p>INPUT</p> <ul style="list-style-type: none"> <li>• Flag Issuer CA known (Value 'No')</li> </ul> <p>OUTPUT</p> <ul style="list-style-type: none"> <li>• OCSF Response (status "unknown")</li> </ul>

		<b>Actions</b> <ul style="list-style-type: none"> <li>Construct an OCSP response with the cert status 'unknown'</li> <li>Sign the OCSP response with key corresponding to the PEPS' certificate used for OCSP signing (C-PEPS certificate and key).</li> <li>Send the OCSP response to the SP.</li> </ul>
2	[Issuer CA is known] Check OCSP support	<b>Sequence Diagram</b> <pre> sequenceDiagram     participant opt     participant line     opt-&gt;&gt;line: [Issuer CA is known]     line-&gt;&gt;line: Check OCSP support   </pre> <b>External Actors</b> <ul style="list-style-type: none"> <li>None</li> </ul> <b>Config files needed</b> <ul style="list-style-type: none"> <li>Issuer CA list</li> </ul> <b>Data</b> <b>INPUT</b> <ul style="list-style-type: none"> <li>Flag Issuer CA known (Value 'Yes')</li> </ul> <b>OUTPUT</b> <ul style="list-style-type: none"> <li>Location of OCSP Responder</li> <li>Flag OCSP support (Value 'Yes')</li> </ul> <b>Actions</b> <ul style="list-style-type: none"> <li>Checks whether the Issuer CA supports OCSP and determine the location (IP address, port number) of the OCSP responder</li> </ul>
3	[Issuer CA supports OCSP] OCSP request	<b>Sequence Diagram</b> <pre> sequenceDiagram     participant opt     participant line     opt-&gt;&gt;line: [Issuer CA supports OCSP] OCSP request   </pre> <b>External Actors</b> <ul style="list-style-type: none"> <li>IDP (i.e. OCSP Responder)</li> </ul> <b>Data</b> <b>INPUT</b> <ul style="list-style-type: none"> <li>Location of OCSP Responder</li> <li>Flag OCSP support (Value 'Yes')</li> </ul> <b>OUTPUT</b> <ul style="list-style-type: none"> <li>OCSP request</li> </ul> <b>Actions</b> <ul style="list-style-type: none"> <li>Forwards the OCSP request received from the colleague PEPS to the OCSP responder</li> </ul>
4-5	[OCSP Responder not available] Error; [OCSP Responder not	<b>Sequence Diagram</b>

	available] Create OCSF response	
		<b>External Actors</b> <ul style="list-style-type: none"> <li>IDP (i.e. OCSF Responder)</li> <li>Colleague PEPS</li> </ul>
		<b>Data</b> INPUT <ul style="list-style-type: none"> <li>Error: IDP is not available</li> </ul> OUTPUT <ul style="list-style-type: none"> <li>OCSF response</li> </ul>
		<b>Actions</b> <ul style="list-style-type: none"> <li>Forwards the OCSF request received from the colleague PEPS to the OCSF responder</li> </ul>
6-7	OCSF Response; Check and re-sign OCSF response	<b>Sequence Diagram</b>
		<b>External Actors</b> <ul style="list-style-type: none"> <li>IDP (i.e. OCSF Responder)</li> <li>Colleague PEPS</li> </ul>
		<b>Config files needed</b> <ul style="list-style-type: none"> <li>C-PEPS certificate and key</li> </ul>
		<b>Data</b> INPUT <ul style="list-style-type: none"> <li>Error “OCSP responder not available”</li> </ul> OUTPUT <ul style="list-style-type: none"> <li>OCSF response, signed by the C-PEPS</li> </ul>
		<b>Actions</b> <ul style="list-style-type: none"> <li>Construct an OCSF response with the cert status ‘unknown’</li> <li>Sign the OCSF response (with the key in “C-PEPS certificate and key” config file).</li> <li>Send the OCSF response to the SP.</li> </ul>
8	[Issuer CA does not support OCSF] Check CRL support	<b>Sequence Diagram</b>



9	[Issuer CA does not support CRL] Create OCSP response	<p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>None</li> </ul> <p><b>Config files needed</b></p> <ul style="list-style-type: none"> <li>Issuer CA list</li> </ul> <p><b>Data</b></p> <p>INPUT</p> <ul style="list-style-type: none"> <li>Flag Issuer CA known (Value 'Yes')</li> </ul> <p>OUTPUT</p> <ul style="list-style-type: none"> <li>Flag CRL support (Value 'Yes' or 'No')</li> <li>Location of CRL</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li>Checks whether the Issuer CA supports CRL (Issuer CA list) and determine the location (e.g. URL) of the CRL</li> </ul> 
		<p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>Colleague PEPS</li> </ul> <p><b>Config files needed</b></p> <ul style="list-style-type: none"> <li>C-PEPS certificate and key</li> </ul> <p><b>Data</b></p> <p>INPUT</p> <ul style="list-style-type: none"> <li>Flag OCSP support (Value 'No')</li> <li>Flag CRL support (Value 'No')</li> </ul> <p>OUTPUT</p> <ul style="list-style-type: none"> <li>OCSP response, cert status "unknown"</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li>Construct an OCSP response with the cert status 'unknown'</li> <li>Sign the OCSP response (with the key in "C-PEPS certificate and key" config file).</li> <li>Send the OCSP response to the colleague PEPS.</li> </ul>
10	[Issuer CA supports CRL] Get CRL	<p><b>Sequence Diagram</b></p>  <p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>IDP</li> </ul>

11	[CRL found] Issue OCSF response	<p><b>Config files needed</b></p> <ul style="list-style-type: none"> <li>• Issuer CA list</li> </ul> <p><b>Data</b></p> <p>INPUT</p> <ul style="list-style-type: none"> <li>• Flag CRL support (Value 'Yes')</li> <li>• Location of CRL</li> </ul> <p>OUTPUT</p> <ul style="list-style-type: none"> <li>• CRL or error message "CRL not found"</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li>• Get the CRL</li> </ul> <p><b>Sequence Diagram</b></p>  <p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>• Colleague PEPS</li> </ul> <p><b>Config files needed</b></p> <ul style="list-style-type: none"> <li>• C-PEPS certificate and key</li> </ul> <p><b>Data</b></p> <p>INPUT</p> <ul style="list-style-type: none"> <li>• CRL</li> </ul> <p>OUTPUT</p> <ul style="list-style-type: none"> <li>• OCSF response</li> </ul> <p><b>Actions</b></p> <ul style="list-style-type: none"> <li>• Construct an OCSF response, in base on the information in the CRL</li> <li>• Sign the OCSF response (with the key in "C-PEPS certificate and key" file).</li> <li>• Send the OCSF response to the colleague PEPS.</li> </ul>
12	[CRL not found] Create OCSF response (status "unknown")	<p><b>Sequence Diagram</b></p>  <p><b>External Actors</b></p> <ul style="list-style-type: none"> <li>• Colleague PEPS</li> </ul> <p><b>Config files needed</b></p> <ul style="list-style-type: none"> <li>• C-PEPS certificate and key</li> </ul> <p><b>Data</b></p> <p>INPUT</p> <ul style="list-style-type: none"> <li>• Error message "CRL not found"</li> </ul>

		<b>OUTPUT</b> <ul style="list-style-type: none"> <li>OCSP response (status “unknown”)</li> </ul>
		<b>Actions</b> <ul style="list-style-type: none"> <li>Construct an OCSP response with the cert status ‘unknown’</li> <li>Sign the OCSP response (with the key in “C-PEPS certificate and key” config file).</li> <li>Send the OCSP response to the colleague PEPS.</li> </ul>

*Table 7 – Description sequence Certificate Validation in C-PEPS*

### 2.3.3 Config files

The PEPS requires some configuration files and lists to be defined.

File	Description
Configuration file	General file that contains information for the PEPS system. Data items: <ul style="list-style-type: none"> <li>Maximum and Minimum QAA level allowed</li> <li>Validation method for SPs</li> <li>Time frame for IP requests</li> <li>Text for Message errors</li> <li>Maximum size for SAML messages</li> </ul>
Colleague list	General list that contains the colleagues PEPS or V-IDP. Data items: <ul style="list-style-type: none"> <li>Country Name</li> <li>Location (IP Address, port number,...) (acting as OCSP Responder)</li> <li>Certificate for signing, including public key to encrypt the messages with the public key of the destination to preserve confidentiality</li> <li>the current status of the X.509 certificate of PEPS used for OCSP signing</li> <li>CV Request URL</li> <li>CV Reply URL</li> </ul>
Issuer CA list	A configuration file that contains: <ul style="list-style-type: none"> <li>a list of known issuer CAs (in C-PEPS’s country), together with the following information for each CA:               <ul style="list-style-type: none"> <li>the certificate of the Issuer CA.</li> <li>the current status of the X.509 certificate of the issuer CA.</li> <li>the country of the CA</li> </ul> </li> <li>the list of Issuer CAs in the colleague PEPS country, together with the hash of the Issuer Name, the hash of the Issuer key and the hash algorithm.</li> </ul>

<p>If OCSF is supported:</p> <ul style="list-style-type: none"> <li>the location (IP address, port number) of the OCSF responder providing responses for the certificates issued by the Issuer CA</li> <li>the X.509v3 certificate of the OCSF Responder (issued by the Issuer CA)</li> <li>the current status of the X.509v3 certificate of the OCSF Responder</li> </ul> <p>If CRL is supported:</p> <ul style="list-style-type: none"> <li>the location of the CRL (URL, or location in the local cache)</li> </ul>
--

Table 8 – Configuration files in PEPS

### 2.3.4 Administration tasks

File	Description
Colleague PEPS list maintenance	Add, delete or modify the list of colleagues that can send and receive Authentication Requests.  <u>Action:</u> Manual

Table 9 – PEPS Administrative tasks

### 2.3.5 Error Handling

Catalogue with the error codes and messages to show to the citizen to inform when an error occurs.

Error	Action	Message
CPAU0101	Validate Request Format	format is not correct
CPAU0102	Validate Origin	origin is not correct
CPAU0103	Check SAML Conditions	conditions are not compliant
CPAU0104	Validate requested QAA Level	max PEPS level is lower than requested QQA level
CPAU0105	Identify source attribute	a mandatory attribute is not available
CPAU0301	Check citizen reply format	response is malformed
CPAU0302	Check consent	consent is not given for a mandatory attribute
CPAU0601	Check AU Request	authentication not successful
CPAU0701	MS Attribute Supply	values for a mandatory attribute not found
CPAU0801	Check AT Verification Request	verification of a mandatory attribute fails
CPAU1401	Check citizen reply format	reply format is malformed
CPAU1402	Check value consents	consent is not given for a mandatory attribute
SPAU0101	Check SP ID and QAA Level	SP ID and QAA Level are missing or max

SPAU0401		PEPS level is lower than requested QAA level
SPAU0102 SPAU0402	Check SP	SP ID is not in SP's list
SPAU0103	Check SP Domain	Origin is not correct
SPAU0403	Check SP Domain	Request domain or Redirect URL is not correct
SPAU0104 SPAU0404	Check number of requests	Request number is greater than or equal to the maximum number allowed.
SPAU0405	Check Selected Country	Selected country is not a valid country
SPAU0406	Check contents	Any mandatory or optional attributes isn't in SP's Attributes list
SPAU2201	Validate Response Format	Format is not correct
SPAU2202	Validate Origin	Origin is not correct
SPAU2203	Check SAML Conditions	Conditions are not compliant

Table 10 – Error Catalogue

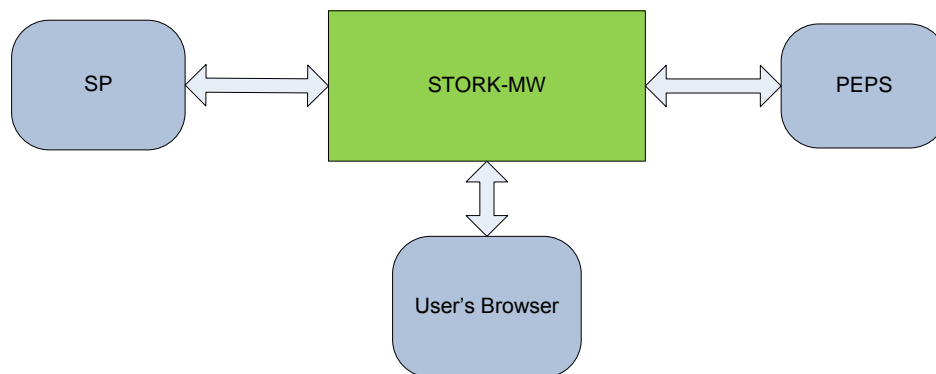
### 3 Middleware

This section deals with the software architecture of the middleware (MW) countries. In Germany and Austria there are already existing MW solutions in place, which must be taken into account. Thus, it is not possible to design a STORK MW system from scratch. Furthermore, the existing systems are quite different in some aspects and therefore it is not possible to define a common STORK MW. The solution described here integrates all existing MW solutions as well as the PEPS solution. The main functional design goals can be summarized as follows:

- Common SP/PEPS interface: The interface used by the SPs must be independent from the user's nationality. This means, no matter what actual authentication mechanism is used (PEPS or MW) the process for the SP is the same. The same interface is also used by PEPS instances to integrate MW solutions.
- Allow for different MW approaches: As the MW solutions might be quite different the V-IDP design must provide country specific processes to deal with this diversity.

#### 3.1 System Context

The following diagram shows the MW system and its interfaces to the environment.



*Figure 9: MW System Context*

The green drawn part of Figure 99 shows the system described here. It is labelled STORK-MW and consists of the V-IDP and (multiple) SPWare components shown later. The main goal of this diagram is to define the system borders.

On one hand the system interacts with the SPs, where the authentication requests and responses are exchanged. On the other end it interacts with PEPS instances. Via this interface the PEPS can request authentication data for MW citizens. The same interface is used to interact with PEPS instances to authenticate a PEPS citizen to the SP of a MW country. Finally, STORK-MW requires user interaction and thus has an interface with the browser.

#### 3.2 Use case view and other requirements

The main functionality of D5.7, which is relevant for the software architecture, is the authentication use case. In particular we are going to analyse the following scenarios:

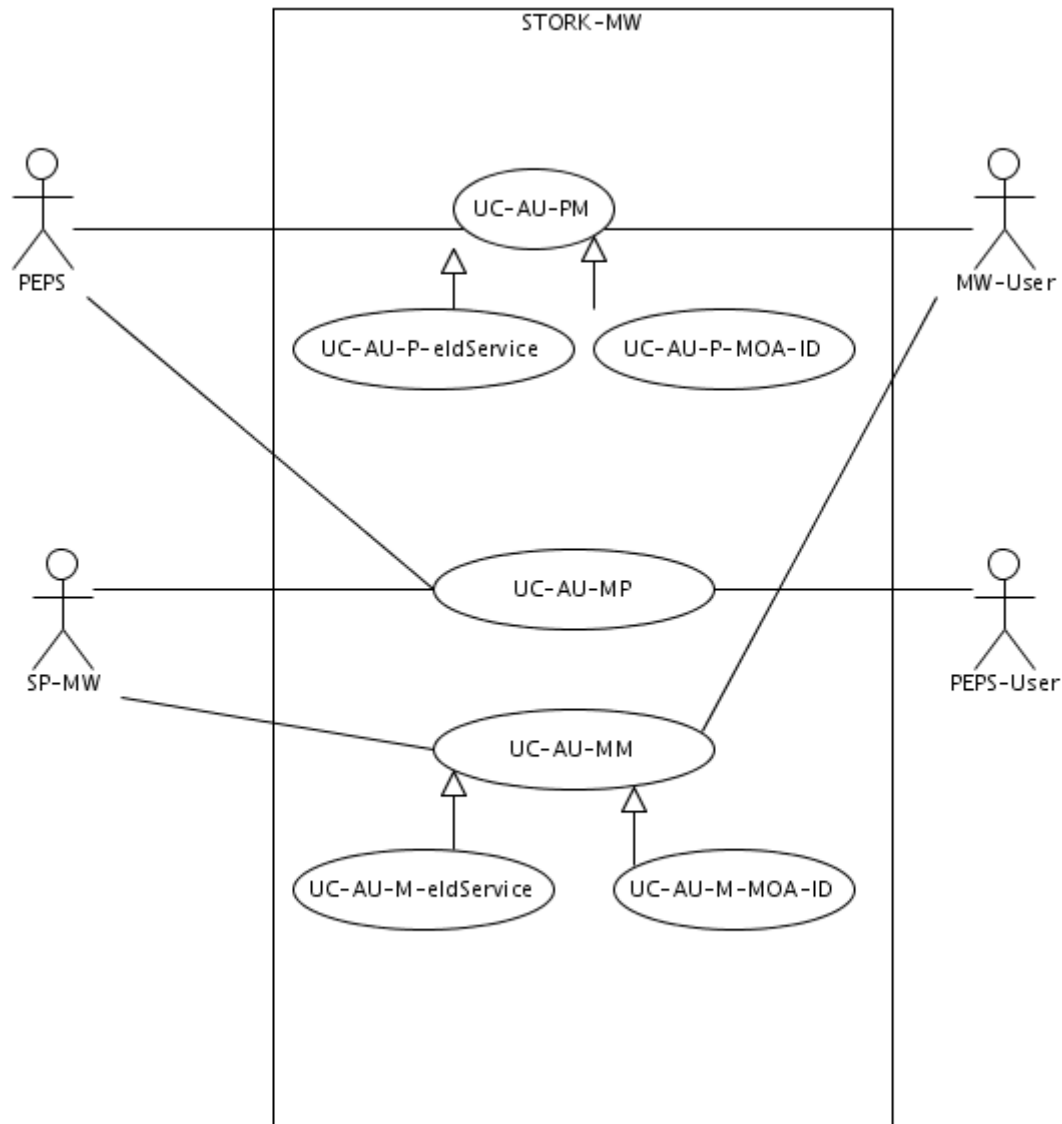
- UC-AU-PM: In this scenario a citizen of a MW country needs to authenticate to use a service in a PEPS country.
- UC-AU-MP: In this scenario a citizen from a PEPS country wants to consume a service from a MW country.

- UC-AU-MM: User presents an eID card from a MW country to a service provider located in a MW country.

For a more detailed analysis of these use cases please refer to D5.7.

### 3.2.1 Use case view

In this section we take the use case definitions of D5.7 as listed above and analyse the MW related parts in more details.



**Figure 10: Use Case Diagram**

The use case diagram above shows the architectural relevant use cases as defined in D5.7. To ensure to be compatible with existing MW solutions we have refined some of the existing use cases. The following use cases are a refinement of D5.7:

- UC-AU-P-eIdService: this is a refinement for UC-AU-PM. In this scenario the German eIdService is used as MW.
- UC-AU-P-MOA-ID: this is a refinement for UC-AU-PM. In this scenario the Austrian MOA-ID is used as MW.

- UC-AU-M-eIdService: this is a refinement of UC-AU-MM. In this scenario the German eIdService is used as MW. The interface to the SP is the same in Germany and Austria. Therefore we do not make a difference where the SP is located and simply use a generic SP-MW.
- UC-AU-M-MOA-ID: this is a refinement of UC-AU-MM. In this scenario the Austrian MOA-ID is used as MW. The interface to the SP is the same in Germany and Austria. Therefore we do not make a difference where the SP is located and simply use a generic SP-MW.

### 3.2.2 Non Functional requirements

- Certificate validation

In order to validate digital certificates, initial implementations of STORK will use the OCSP protocol. For V-IDP implementations it should be possible to exchange validation messages with S-PEPS and C-PEPS implementations. The support of OCSP is optional for V-IDPs and will not be supported in the reference implementation for the pilots.

### 3.3 Logical view

The following diagram shows the components, which make up the STORK-MW (drawn in green) and the external components (drawn in white).

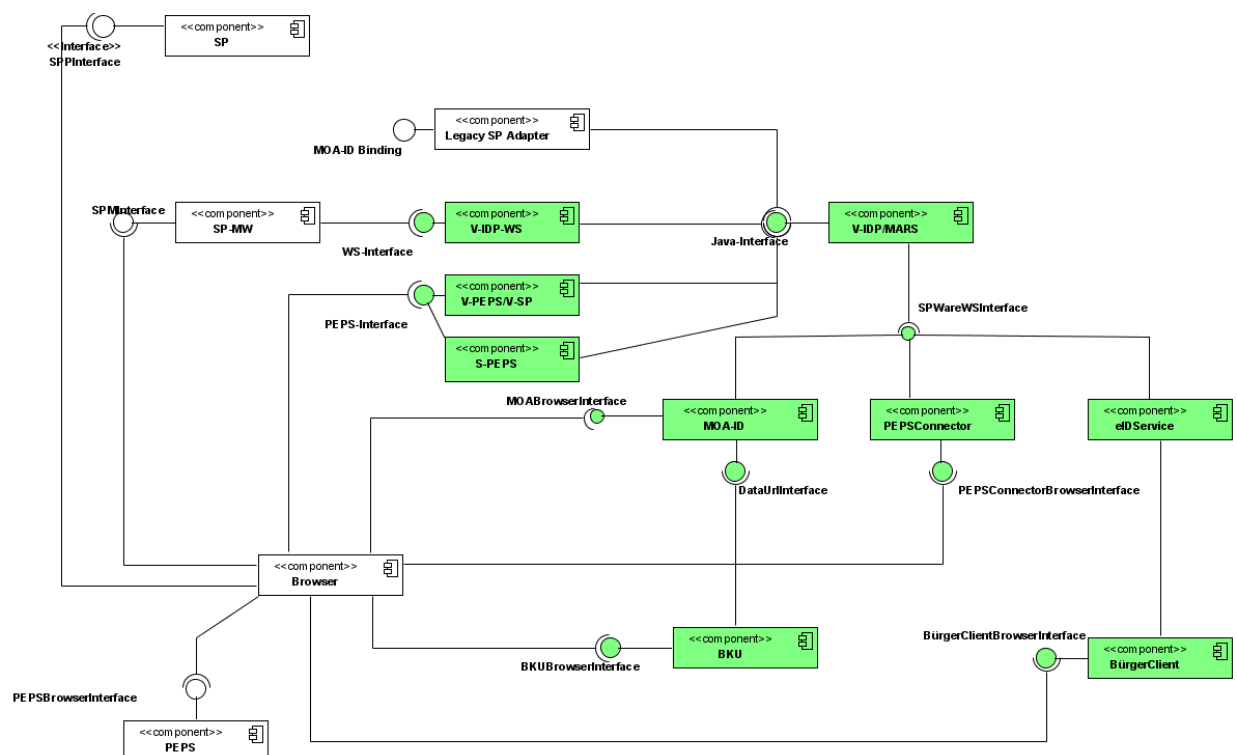





























Figure 11: STORK-MW Components

Name	Documentation
 SPPIInterface	The web interface for a service provider located in a PEPS country (SP).
 SP-PEPS	The service provider located in a PEPS country.



 Legacy SP Adapter	This component is just an example how to integrate legacy service providers. Every MW country can implement its own adapter to map the country specific request (binding) onto the common Java software interface.
 MOA-ID Binding	This binding defines the current binding of the Austrian SPWare MOA-ID.
 SP-MW	The service provider located in a MW country.
 V-IDP-WS	This component is the implementation of the web service interface (WS-Interface). Its main purpose is the handling of the SOAP binding and the mapping of the exchanged messages to the Java Interface.
 V-IDP/MARS	The virtual identity provider implements the common Java-Interface and also offers a common interface for different SPWares. Its main duty is the routing of the requests coming from a PEPS or SP-MW to the according SPWare implementation.
 SPMInterface	The web interface of SP-MW.
 WS-Interface	This interface offers a web service (SOAP) binding for service providers (SP-MW).
 Java-Interface	This is the common software interface based on Java. On top of it several protocol/binding adapters are placed to deal with PEPS, SP-MW and legacy service providers.
 V-PEPS/V-SP	This implementation of the PEPS-Interface handles the POST binding and maps the incoming request to the Java-Interface. The V-PEPS/V-SP represents logically the S-PEPS within the V-IDP component.
 PEPS-Interface	This interface defines the interface used by PEPS instances. It uses a HTTP POST binding.
 SPWareWSInterface	All SPWare implementations are expected to have a common webservice interface.
 S-PEPS	The S-PEPS component of the V-IDP/MARS offers all necessary interfaces and functions for providing a S-PEPS service.
 MOA-ID	This component defines the SPWare of Austria. It encapsulates the server component used for the Austrian authentication process. The current MOA-ID implementation does not have the features described here. Thus the MOA-ID shown here is the future (STORK enabled) MOA-ID component.
 PEPSConnector	This SPWare component is responsible for handling the communication with all PEPS instances (C-PEPS). All PEPS specific functionality from a V-IDP's point of view is encapsulated.
 eIDService	The server component used for the German authentication process.
 MOABrowserInterface	The web interface of MOA-ID to communicate with the user's browser. It is required for BKU selection and to start the authentication process.
 DataUrlInterface	This interface is used by the BKU to communicate directly with MOA-ID.
 PEPSConnectorBrowserInterface	This web interface is used to communicate with the user's browser. The PEPS POST binding is realized via this interface.

 Browser	The user's browser.
 BKU	The citizen card environment (Bürgerkartenumgebung) used in Austria.
 BKUBrowserInterface	This interface used by the BKU to communicate with the user's browser.
 BürgerClient	The citizen smart card environment of the German eCardAPI running on the client PC.
 BürgerClientBrowserInterface	This interface used by the BürgerClient to communicate with the user's browser.
 PEPSBrowserInterface	The web interface of the PEPS used to send and receive (authentication) messages.
 PEPS	The Pan European Proxy Service as used in STORK. For each PEPS member state one instance exists.

*Table 11 – Stork MW components description*

### 3.3.1 Use Case Analysis

The following sub-sections analyse the use cases of chapter 3.2.1 and explains the responsibilities of the components listed above.

#### 3.3.1.1 UC-AU-P-MOA-ID

This use case describes the situation where an Austrian citizen wants to consume a service offered by a PEPS country. For the authentication process the Austrian SPWare (MOA-ID) is used. This situation is shown in the following diagram.

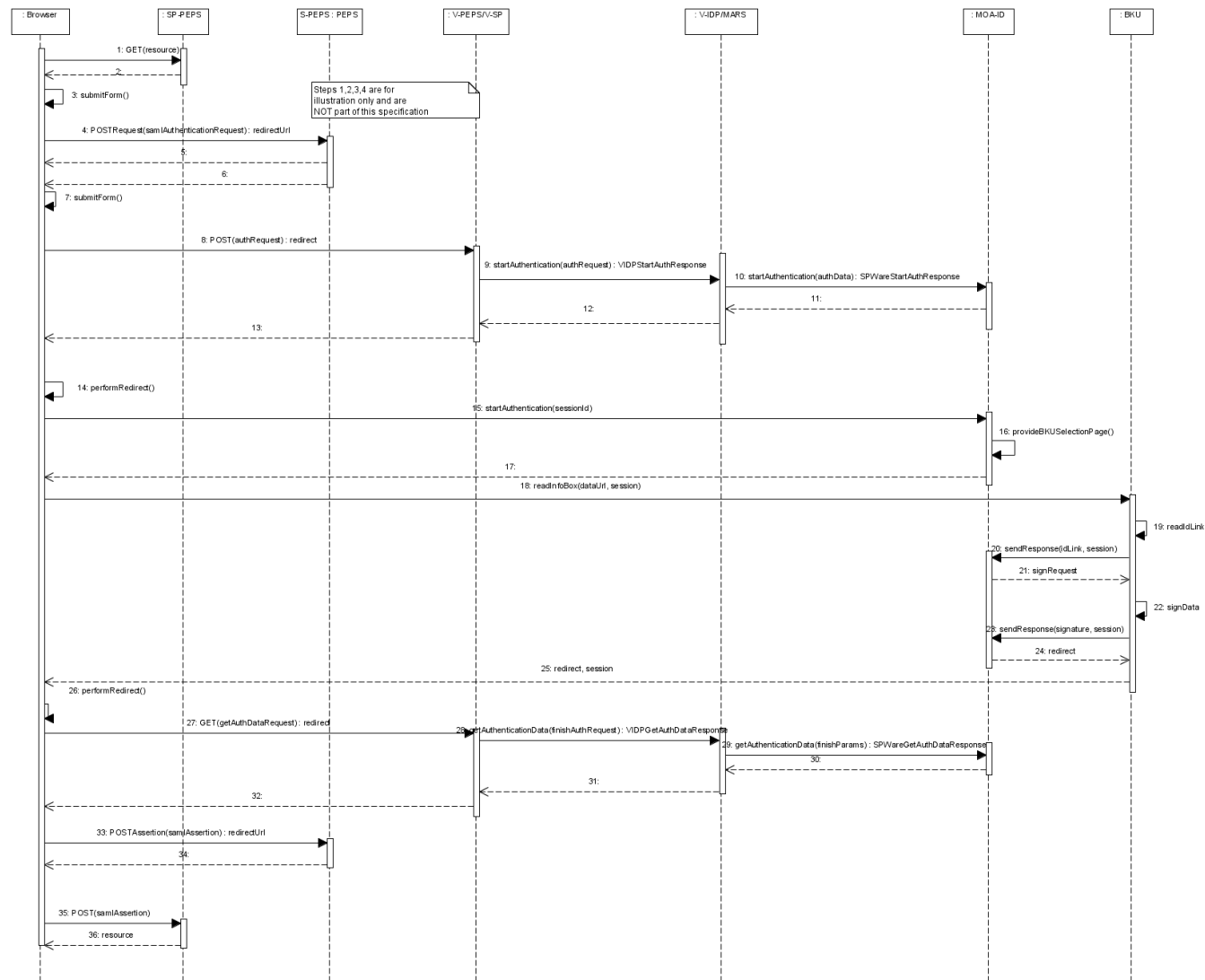


Figure 12:UC-AU-P-MOA-ID

Step	Description
1,2,3,4	A citizen from a MW country wants to use a service offered by a SP in a PEPS country. The SP-PEPS requires the user to authenticate and thus sends an authentication request via the user's browser to the PEPS. The PEPS recognizes the authentication request as a request from a MW citizen. The request is transformed into a "common format" (SAML AuthnRequest). These steps are for illustration purposes only and may miss some steps like e.g. the country selection. The realization of these steps can differ from country to country and is NOT part of this specification.
5,6,7,8	The PEPS sends an authentication request to the V-IDP. The interface used for sending this authentication request is identical to a PEPS interface and specified there in detail. The V-PEPS/V-SP component handles the specified HTTP POST binding and hands over the data to the V-IDP using a Java interface. The relevant data is defined in the class VIDPStartAuthRequest. The V-IDP analyzes the CCC and looks up the relevant SPWare "connector" (in this use case MOA-ID).
9	V-IDP tells MOA-ID to start the authentication process. The parameters are defined in class SPWareStartAuthRequest. MOA-ID generates a session for this request.
10	MOA-ID returns the URL to its own HTTP web interface (MOABrowserInterface). The URL contains the previously generated session id.
11,12	The V-IDP forwards the URL to the V-PEPS/V-SP which sends a HTTP redirect back to the user's browser.
13,14	The browser performs the redirect and contacts MOA-ID's web interface by sending the session Id encoded in the redirect URL.
15,16	MOA-ID offers a page with potentially multiple HTML forms for different implementations of the citizen card environment (BKU). Each form contains commands for the BKU to read out the IdLink from the citizen card.
17	The user selects one BKU variant and the data is sent via HTTP POST to be BKU.
18,19	The BKU interprets the commands (reads out the IdLink data) and sends the result via the dataUrl back to MOA-ID.
20	MOA-ID checks the received data and generates the according signature request.
21,22	The BKU requests the user to sign the data and sends back the signature result to MOA-ID.
23, 24, 25	MOA-ID validates the signature and causes the browser to redirect to the V-PEPS/V-SP (the redirect URL must contain the VIDPGetAuthDataRequest parameters encoded (session id and CCC)).
26	The browser sends the data to the V-PEPS/V-SP via a GET request to finish the authentication.
27	The V-PEPS/V-SP forwards the data to the V-IDP.
28	The V-IDP determines the SPWare matching the encoded CCC and calls the corresponding SPWare method.
29	MOA-ID transforms the Austrian SAML construct read from the citizen card into a STORK compliant format, digitally signs it and sends it back to the V-IDP.
30,31,32	The V-IDP sends the received SAML assertion to the V-PEPS/V-SP, which in turn sends back a HTML form to the browser causing the browser to POST the SAML assertion to the requesting S-PEPS.

33, 34,35	<p>The S-PEPS transforms the assertion into a member state specific format and causes the browser to send this new assertion to the SP-PEPS which in turn returns the resource requested in step 1.</p> <p>These steps are NOT scope of this specification.</p>
-----------	---

*Table 12 – Authentication process for AT citizens accessing SP of a PEPS country*

### 3.3.1.2 UC-AU-P-eIdService

This uses case describes the situation where a German citizen wants to consume a service offered by a PEPS country. For the authentication process the German MW (eIdService) is used. This situation is shown in the following diagram.

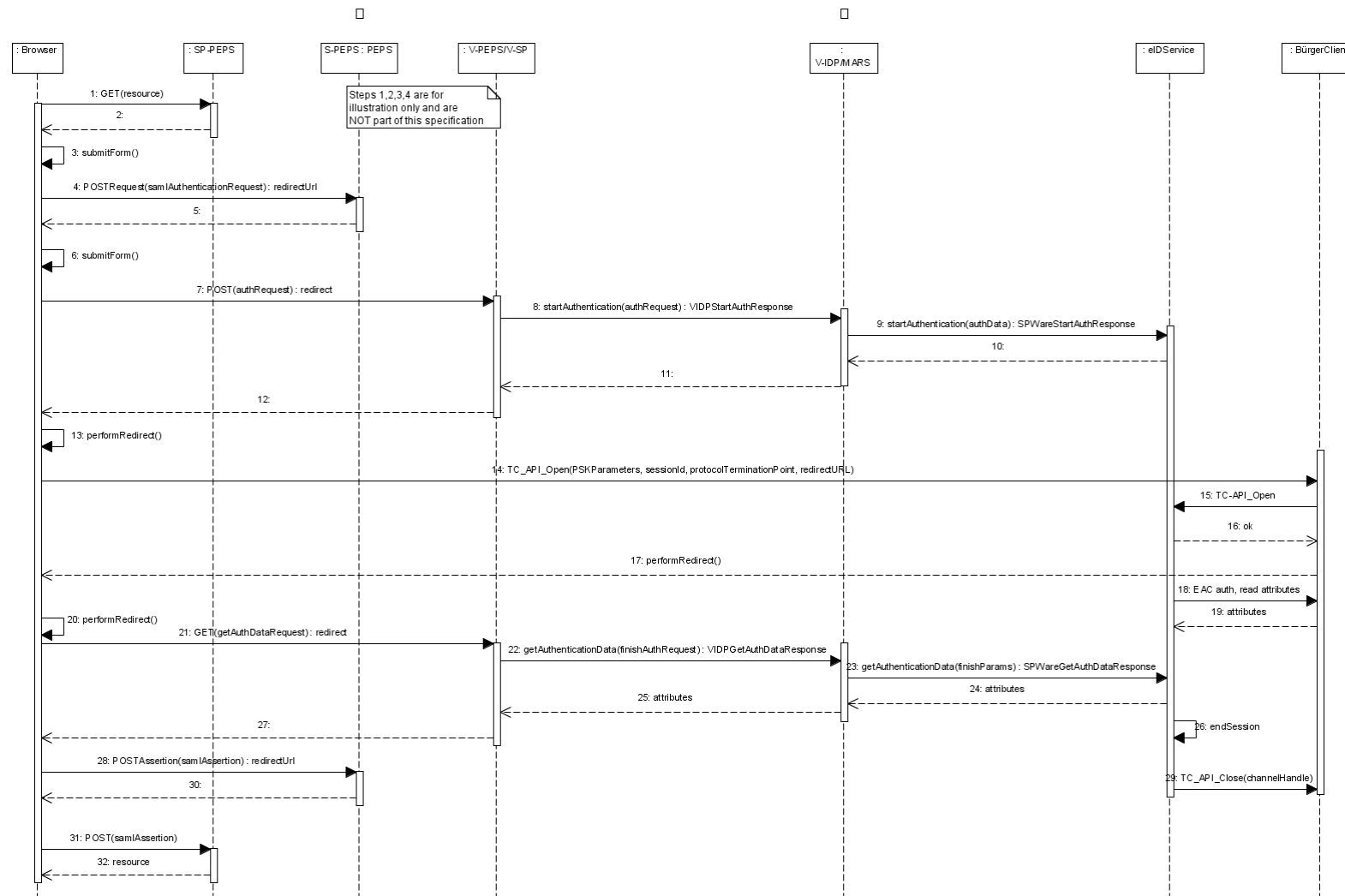


Figure 13: UC-AU-P-eIDService

Step	Description
1,2,3,4, 5, 6	A citizen from a MW country wants to use a service offered from a SP in a PEPS country. The SP-PEPS requires the user to authenticate and thus redirects the browser to the S-PEPS. The S-PEPS recognizes the authentication request as a request from a MW citizen. The request is transformed into a “common format” (SAML AuthnRequest). These steps are for illustration purposes only and may miss some steps like e.g. the country selection. The realization of these steps can differ from country to country and is NOT part of this specification.
7	The S-PEPS sends an authentication request to the V-PEPS/V-SP. The interface is designed according to the S-PEPS interface specification.
8	The V-PEPS/V-SP forwards the authentication request to the V-IDP using the V-IDP Java interface.
9	The V-IDP tells the eIDService to start the authentication process. The parameters are defined in the class SPWareStartAuthRequest. The eIDService generates a session for this request.
10	The eIDService returns the URL for the BürgerClient activation. The URL includes the session id, the protocolTerminationPoint and the TLS pre shared key (PSK) parameters for connection establishment of the eCardAPI client and server instances.
11	The V-IDP returns the result to the V-PEPS/V-SP.
12	The V-PEPS/V-SP sends a HTTP redirect back to the user's browser. The redirect URL contains the sessionid and the pre shared key.
13,14	The browser performs the redirect and contacts the BürgerClient web interface sending the session Id, PSK parameters and protocolTerminationPoint (encoded in the redirect URL).
15,16	The BürgerClient establishes with the help of the session id, the protocolTerminationPoint and the PSK a web service connection with the eIDService.
17	The BürgerClient returns a REDIRECT response ( the redirect URL must contain the VIDPGetAuthDataResponse parameters encoded).
18,19	The BürgerClient and the eIDService are performing the authentication with the ePA and the retrieval of the attributes via the established web service channel.
20, 21	The browser performs a redirect and sends a GetAuthDataRequest to the V-PEPS/V-SP (special URL with session id encoded).
22	The V-PEPS/V-SP sends a GetAuthDataRequest to the V-IDP.
23	The V-IDP determines the corresponding SPWare and forwards the request to the eIDService in this case.
24	The eIDService transforms the attributes into a STORK compliant format, digitally signs and encrypts it for the requesting SP and sends it back to the V-IDP.
25-31	The V-IDP transmits the received attributes via the V-PEPS/V-SP to the S-PEPS, which in turn transforms the data to a member state specific format and hands the authentication data to the SP-PEPS. The realization of these steps can differ from country to country and are NOT part of this specification.
32	The SP-PEPS delivers - after a successful authentication - the requested resource to the Browser.

**Table 13 –Authentication process for DE citizens accessing SP of a PEPS country**

### 3.3.1.3 UC-AU-MP

This use case describes the situation where a citizen of a PEPS country wants to consume a service offered by a MW country. For the authentication process the PEPS is used. As the SP-interface is the same in Germany and Austria we do not further refine this use case but use the generic SP-MW. This situation is shown in the following diagram.



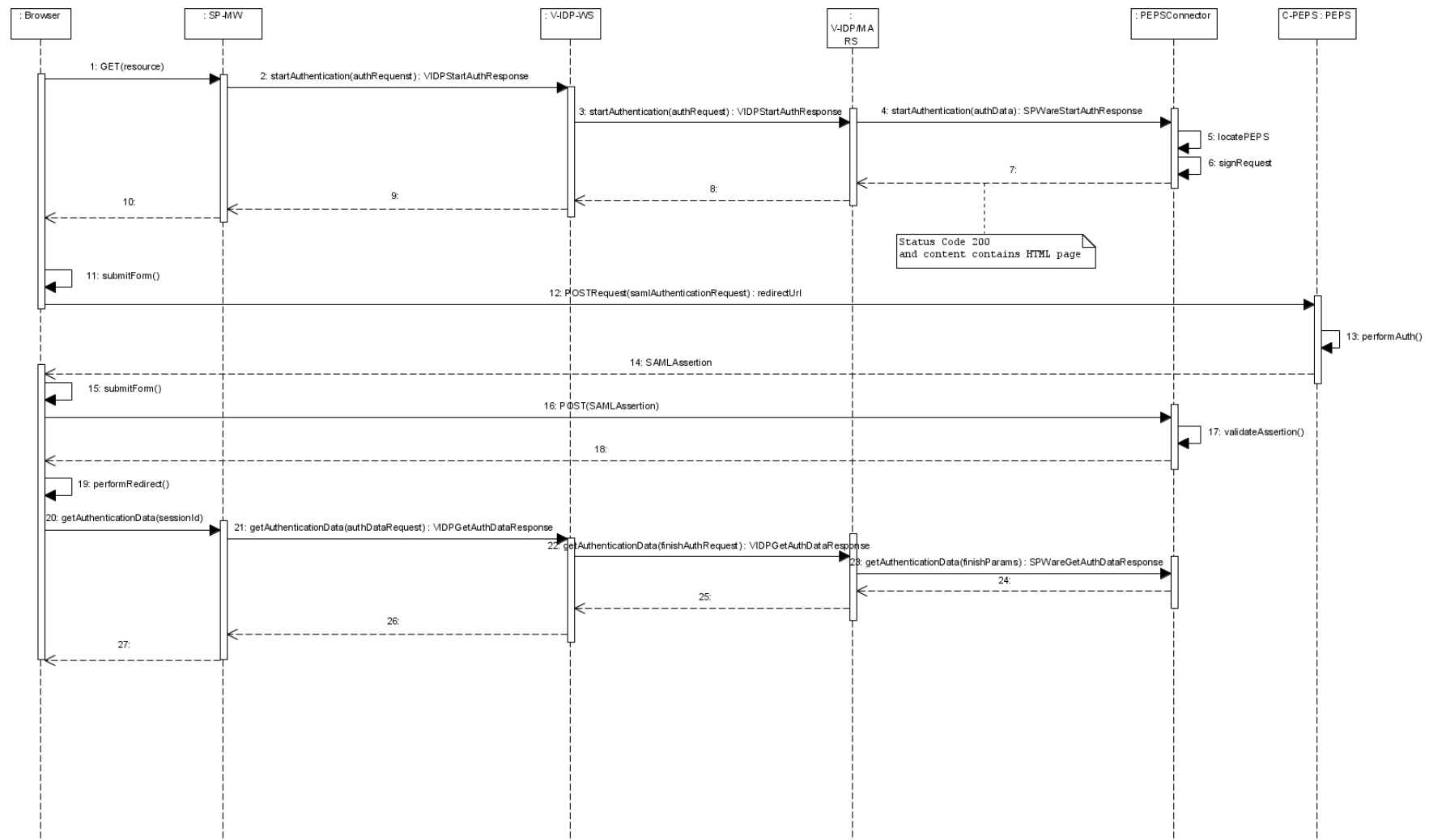


Figure 14: UC-AU-MP

Remark: The SP-MW uses one particular V-IDP instance. This might be a dedicated V-IDP instance for only this SP-WM or one installation shared between multiple service providers.

Step	Description
1	A citizen from a PEPS country wants to use a service offered by a SP located in a MW country (SP-MW). The service provider requires the user to authenticate. To do this, the SP-MW must ask the user for her nationality (not shown in the diagram).
2,3	The SP-MW starts the authentication process by sending a startAuthentication message via the web service interface V-IDP-WS to the V-IDP. The V-IDP looks up the SPWare according to the received CCC. In this use case the PEPSCConnector SPWare is used.
4, 5,6,7,8,9,10	The V-IDP sends a startAuthentication request to the PEPSCConnector which looks up the corresponding PEPS and signs the authentication request. The resulting HTML Page is sent back to the browser.
11, 12	The browser submits the received html form (typically Javascript is used to automatically submit the form) and the data is sent to C-PEPS via HTTP POST.
13, 14, 15,16	The PEPS performs the member state specific authentication process. The resulting SAML assertion is returned (using again the POST binding) to the PEPSCConnector.
17	The PEPSCConnector validates the received SAML assertion.
18,19,20	The user is redirected to the SP-MW.
21,22,23,24,25,26	The SP-MW gets the authentication data from the V-IDP using the web service interface (V-IDP-WS). The V-IDP gets the SAML assertion from the PEPSCConnector and the assertion is sent back to the SP-MW which grants/denies access to the resource the user requested in step 1.
27	The resource is sent to the user's browser.

*Table 14 –Authentication process for citizens of PEPS country accessing SP of a MW country*

### 3.3.1.4 UC-AU-M-MOA-ID

This use case describes the situation where an Austrian citizen wants to consume a service offered by a MW country. For the authentication process the Austrian MW (MOA-ID) is used. As the SP-interface is the same in Germany and Austria we do not further refine this use case but use the generic SP-MW. This situation is shown in the following diagram.

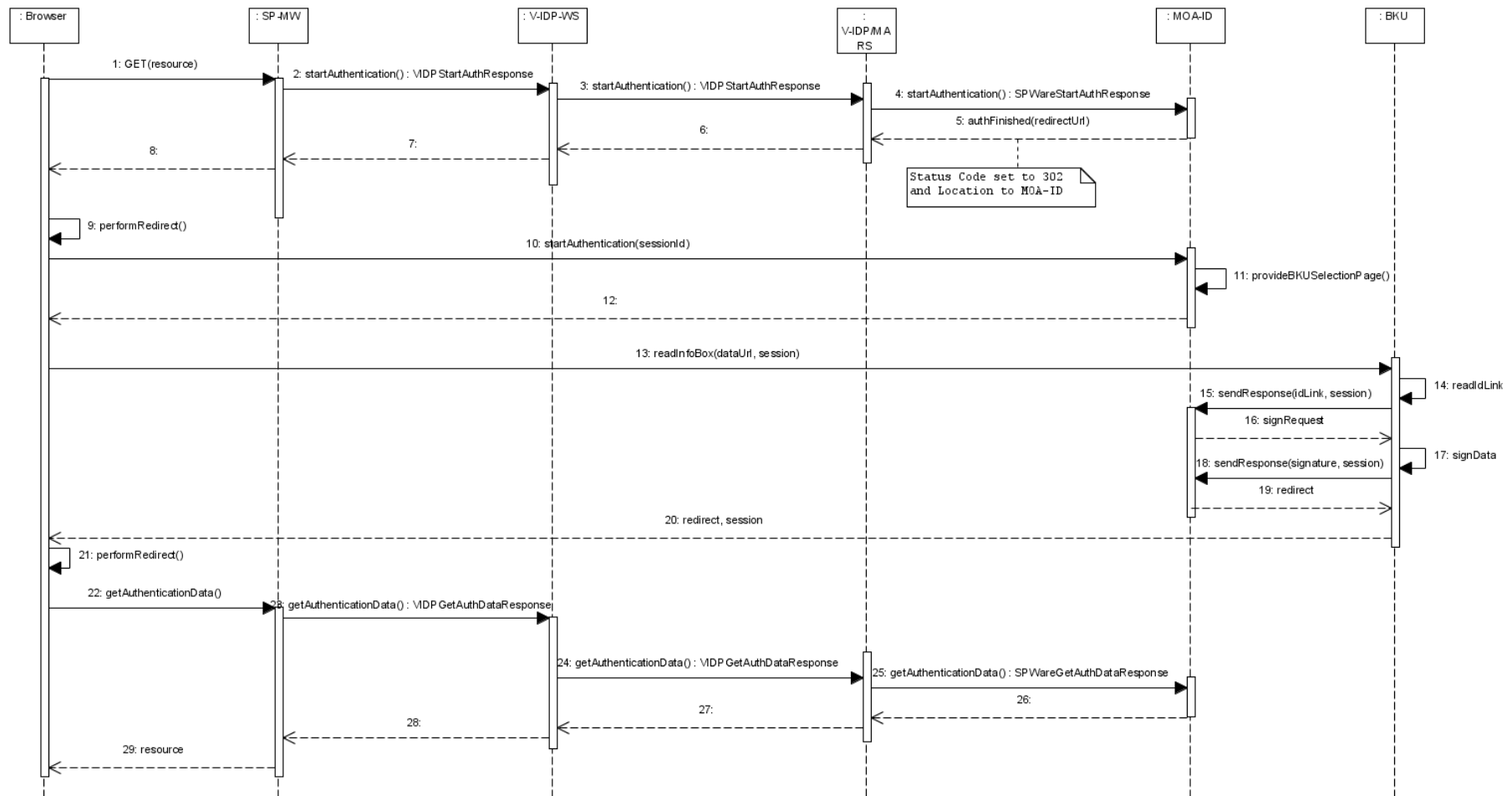


Figure 15: UC-AU-M-MOA-ID

Step	Description
1	<p>A citizen from a MW country wants to use a service offered by another MW country. There will be no difference if the citizen and the service provider are from the same country or from different ones. The SP-MW requires the user to authenticate.</p> <p>Remark: Step 1, is out of scope of this specification and might require additional preceding actions, like country selection.</p>
2	The SP-MW uses the web service interface to authenticate the user. It sends an authentication request to the V-IDP-WS interface. The relevant data is defined in class VIDPStartAuthRequest.
3	The web service interface just exposes the Java interface as web service. This means the same data (VIDPStartAuthRequest) is forwarded to the V-IDP, where the CCC is analyzed and the relevant SPWare (in this case MOA-ID) is looked up.
4	The V-IDP tells MOA-ID to start the authentication process. The parameters are defined in class SPWareStartAuthRequest. MOA-ID generates a session for this request.
5,6,7,8	MOA-ID returns the URL to its own HTTP web interface (MOABrowserInterface). The URL contains the encoded, previously generated session id. This data is returned the call hierarchy back from MOA-ID to the V-IDP, and via the web service (V-IDP-WS) to the SP-MW and finally to the browser.
9,10	The browser performs the redirect and contacts the MOA-ID's web interface sending the session Id (encoded in the redirect URL).
11,12	MOA-ID offers a page with potentially multiple HTML forms for different implementations of the citizen card environment (BKU). Each form contains commands for the BKU to read out the IdLink from the citizen card.
13	The user selects one BKU variant (pressing the relevant button) and the data is sent via HTTP POST to be BKU.
14,15	The BKU interprets the commands (reads out the IdLink data) and sends the result via the dataUrl to MOA-ID.
16	MOA-ID checks the received data and generates the according signature request.
17,18	The BKU requests the user to sign the data and sends back the signature to MOA-ID.
19,20,21	MOA-ID validates the signature and causes the browser to redirect to the SP-MW (the redirect URL must contain the VIDPGetAuthDataRequest parameters encoded).
22	The browser sends a GetAuthenticationData request to the SP-MW.
23,24	SP-MW uses the web service interface (V-IDP-WS) to send the VIDPGetAuthDataRequest to the V-IDP.
25	The V-IDP determines the SPWare matching the encoded CCC and calls the corresponding SPWare method.
26	MOA-ID transforms the Austrian SAML construct read from the citizen card into a STORK compliant format, digitally signs it and sends it back to the V-IDP.
27,28,29	The V-IDP sends the received SAML assertion to SP-MW, which checks it and returns

	the protected resource requested in step 1.
--	---

*Table 15 –Authentication process for citizens of PEPS country accessing AT-SP*

### 3.3.1.5 UC-AU-M-eIdService

This use case describes the situation where a German citizen wants to consume a service offered by a MW country. For the authentication process the German MW (eIdService) is used. As the SP-interface is the same in Germany and Austria we do not further refine this use case but use the generic SP-MW. This situation is shown in the following diagram.

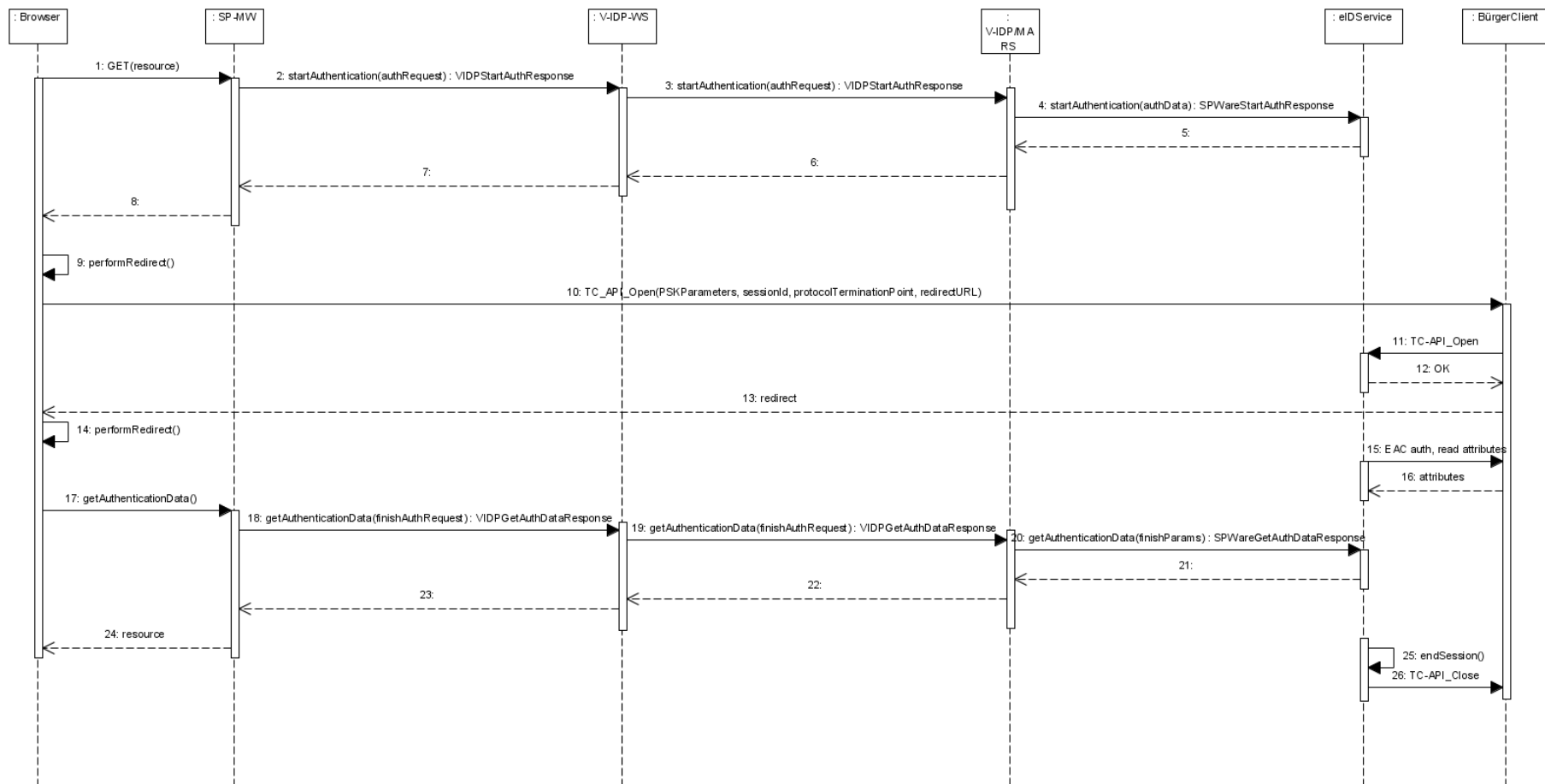


Figure 16:UC-AU-M-eIdService

Step	Description
1	A citizen from a MW country wants to use a service offered by another MW country. There will be no difference if the citizen and the service provider are from the same country or from different ones. The SP-MW requires the user to authenticate and thus sends a startAuthenticationRequest to the V-IDP Remark: Step 1, is out of scope of this specification an might require additional preceding actions, like country selection.
2, 3	The SP-MW sends an authentication request via the V-IDP web service interface to the V-IDP. The relevant data is defined in class VIDPStartAuthRequest. The V-IDP analyzes the CCC and looks up the relevant SPWare (in this use case eIDService).
4	The V-IDP tells the eIDService to start the authentication process. The parameters are defined in the class SPWareStartAuthRequest. The eIDService generates a session for this request.
5	The eIDService returns the URL for the BürgerClient activation. The URL includes the session id, the protocolTerminationPoint and the TLS pre shared key (PSK) parameters for connection establishment of the eCardAPI client and server instances.
6,7	The V-IDP returns the result to the SP-MW.
8	The SP-MW sends a HTTP redirect back to the user's browser. The redirect URL contains the sessionid and the pre shared key.
9, 10	The browser performs the redirect and contacts the BürgerClient web interface sending the session Id, PSK parameters and protocolTerminationPoint (encoded in the redirect URL).
11, 12	The BürgerClient establishes with the help of the session id, the protocolTerminationPoint and the PSK a web service connection with the eIDService.
13,14	The BürgerClient returns a REDIRECT response (the redirect URL must contain the VIDPGetAuthDataResponse parameters encoded).
15,16	The BürgerClient and the eIDService are performing the authentication with the ePA and the retrieval of the attributes via the established web service channel.
17	The browser sends a GetAuthenticationDataRequest to the SP-MW.
18, 19	The SP-MW sends a GetAuthenticationDataRequest to the V-IDP.
20	The V-IDP determines the corresponding SPWare and forwards the request to the eIDService in this case.
21	The eIDService transforms the attributes into a STORK compliant format, digitally signs and encrypts it for the SP-MW and sends it back to the V-IDP.
22, 23	The V-IDP sends the received attributes to the SP-MW.
24	The SP-MW grants or denies access to the requested resource.
25, 26	The eIDService ends the session and closes the connection with the BürgerClient.

**Table 16 –Authentication process for citizens of PEPS country accessing DE SP**

### 3.3.1.6 UC-AU-SPEPS

This use case describes the realization of a SPEPS authentication using V\_IDP/MARS technology.

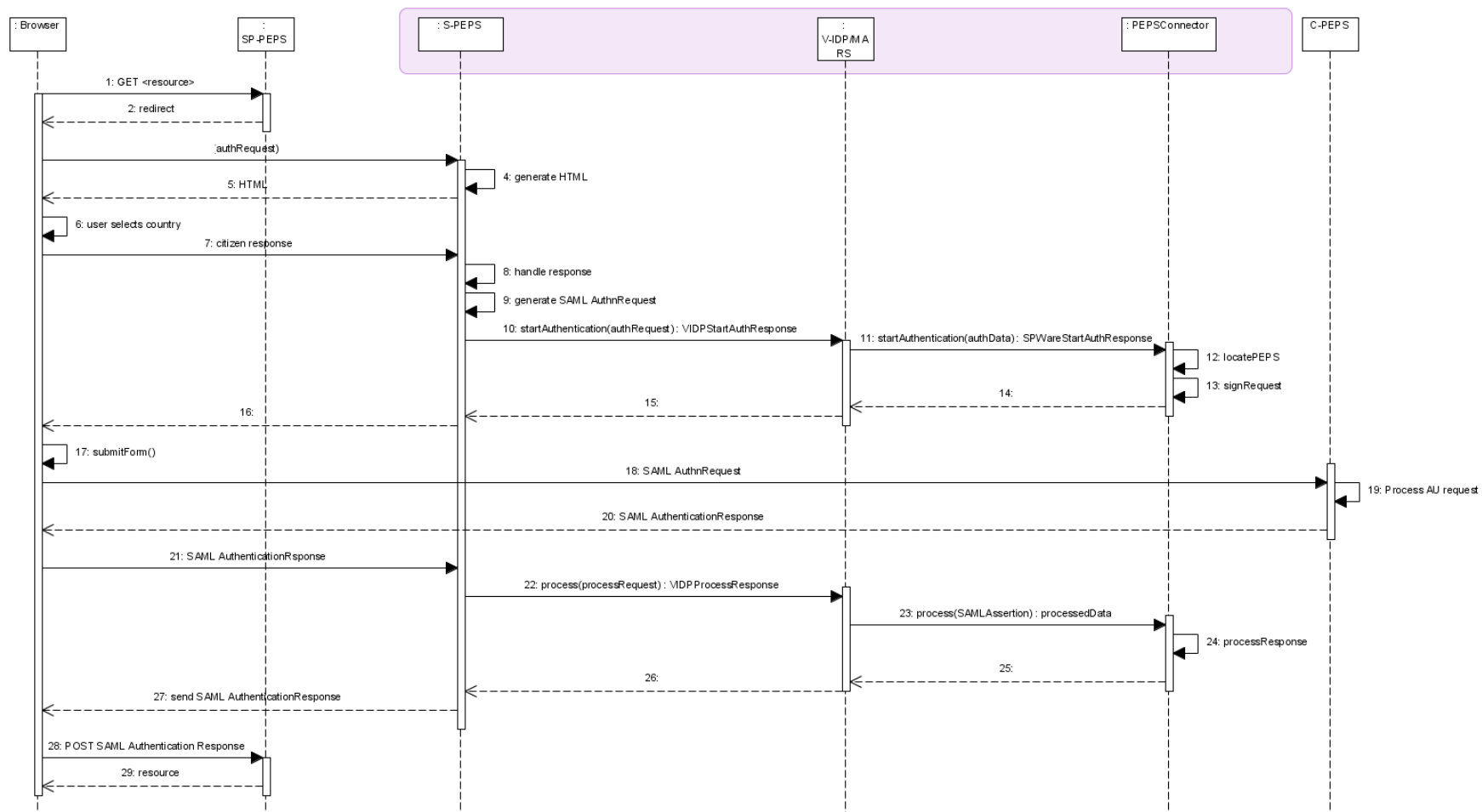


Figure 17 -UC-AU-SPEPS



Step	Description
1,2, 3	A citizen wants to use a service offered from a SP in a PEPS country. The SP-PEPS requires the user to authenticate and thus redirects the browser to the S-PEPS.  The realization of these steps can differ from country to country and is NOT part of this specification.
4,5	The S-PEPS component generates the HTML code for the country selection and returns it to the user agent.
6, 7, 8, 9	The user selects the country. The response is transferred to the S-PEPS, which creates from the original member state specific SP authentication request a standard V-IDP/MARS authentication request for the destination PEPS.
10, 11, 12, 13, 14	The request is forwarded via the V-IDP/MARS component to the PEPS connector, which creates a HTML form with the authentication request and returns it to the user agent. Additionally, the PEPS connector has to locate the destination PEPS and has to sign the authentication request.
15, 16, 17, 18	The SAML AuthnRequest is posted to the C-PEPS. The authentication is (member specific) performed and the resulting SAML assertion is returned.
19	The SAML authentication response is sent to the S-PEPS.
20, 21, 22, 23, 24	The S-PEPS delegates the processing of the response via the V-IDP/MARS component to the PEPS connector which returns the results to the S-PEPS.
25, 26	The authentication response is forwarded to the SP-PEPS.
27	After successful verification, the SP-PEPS returns the requested resource.

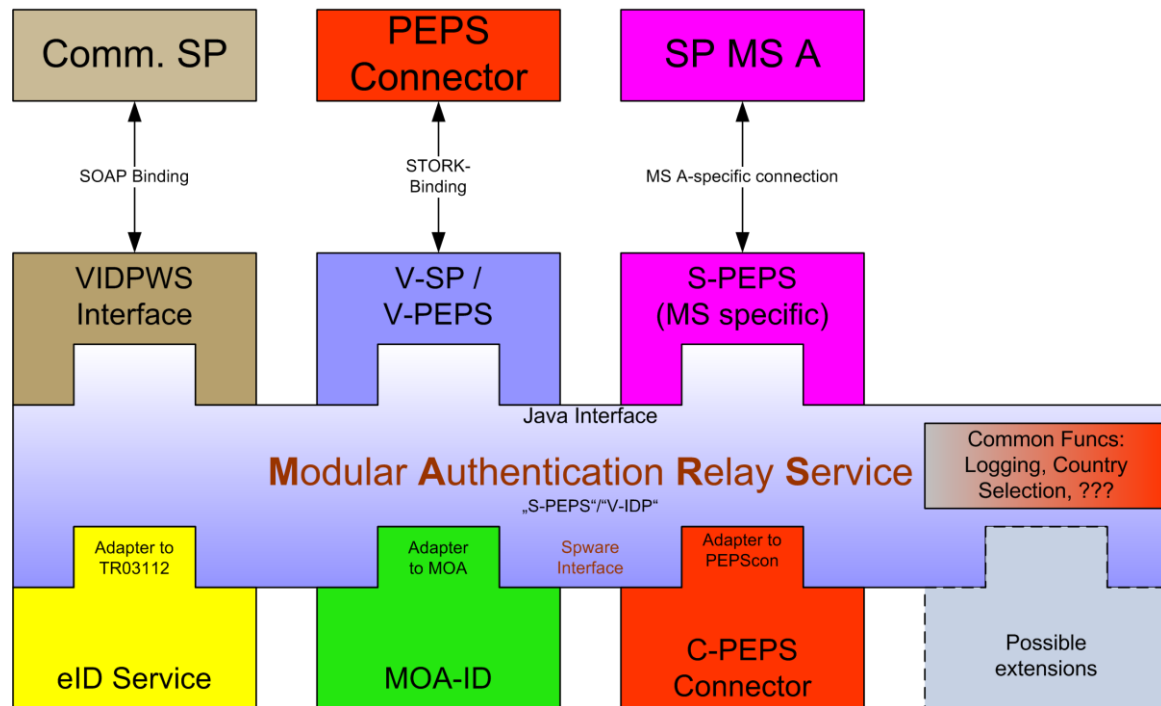
*Table 17 –Authentication process for V-IDP with MARS technology*

### 3.4 Deployment view

The MARS architecture allows for different deployments depending on the usage. It can be deployed in a way that the following functionality can be ensured:

1. V-IDP to sit besides an S-PEPS thus realizing access to MW solutions for an S-PEPS. For this the V-SP/V-PEPS plug-on is needed as well as the MW SPwares.
2. V-IDP to be accessed directly by a SP, thus enabling the SP to perform authentication of foreign eIDs. For this the VIDP Webservice Plug-On is needed as well as all available SPwares including the PEPS connector.
3. S-PEPS. The full functionality of an S-PEPS can be realized using an appropriate plug-on and the PEPS connector as plug-in
4. Legacy eID Service: in some countries legacy SP might expect a certain interface for performing authentication. This is similar to the S-PEPS use case. In this case an appropriate legacy plug-on is needed as well as the necessary SPwares.

The different deployment possibilities are summarized in the following diagram:



*Figure 18 Deployment Options for MARS architecture*

### 3.5 Data view

In this section we analyze the data that has to be sent over the interfaces to complete the processes defined above. The exact data types and their encodings are defined in the interface specification.

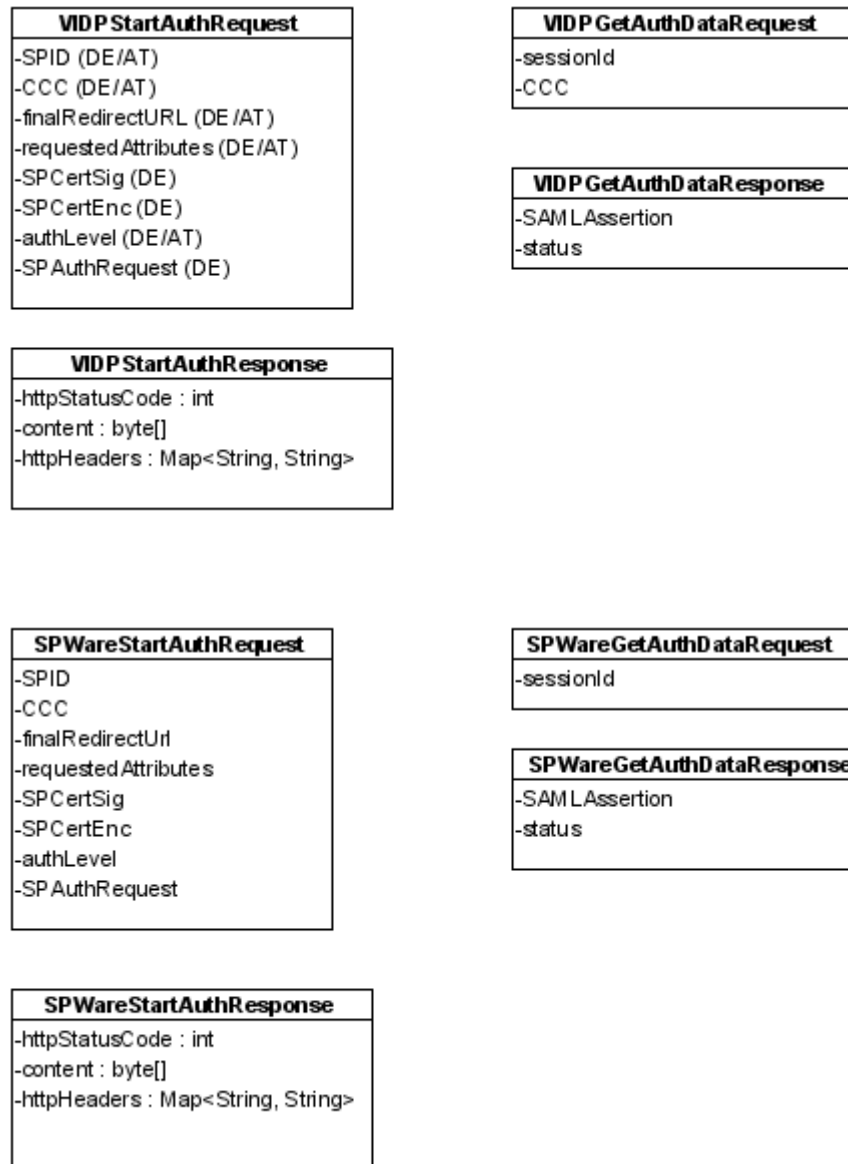


Figure 19: Data model

### 3.5.1 Class VIDPStartAuthRequest

This message is sent to the V-IDP to start an authentication process.

Attributes	Description
SPID (DE/AT)	This attribute uniquely identifies a service provider.
CCC (DE/AT)	The citizen country code to determine the SPWare to be used.
finalRedirectUrl (DE/AT)	The URL to which the browser should be redirected after the authentication process.
requestedAttributes (DE/AT)	The attributes which are requested by a SP.
SPCertSig (DE)	Certificate for the signature verification key of the SP. The SAML AuthnRequest is signed by the SP and the signature can be verified by using this certificate.
SPCertEnc (DE)	The attributes are returned encrypted. For encryption, the key of

	this certificate is used.
authLevel (DE/AT)	Requested authentication level.
SPAuthRequest (DE)	If the VIDPStartAuthRequest is sent from a PEPS this attribute contains the original authentication request as sent from the SP. It is used to authenticate the SP to the eIdService.

*Table 18 – Attributes of VIDPStartAuthRequest*

### 3.5.2 VIDPGetAuthDataRequest

This request is send to the V-IDP to request the attributes from the authentication process associated with the given session id.

Attributes	Description
sessionId	The session to uniquely identify an authentication process within one SPWare. This session is created by the according SPWare.
CCC	The citizen country code to determine the SPWare to be used.

*Table 19 – Attributes of VIDPGetAuthDataRequest*

### 3.5.3 VIDPGetAuthDataResponse

This data structure describes the response data to the VIDPStartAuthRequest.

Attributes	Description
SAMLAAssertion	The SAML assertion, which is the result of the authentication process.
status	The status code, identifying the result of the authentication process.

*Table 20 – Attributes of VIDPGetAuthDataResponse*

### 3.5.4 VIDPStartAuthResponse

This data structure describes the response data to the VIDPStartAuthRequest.

Attributes	Description
httpStatusCode (DE/AT)	The HTTP Status code sent back to the user's browser.
content (DE/AT)	Contains the HTML content. May be null when the response does not contain any content.  The required parameters must be encoded either in the redirect url or in the POST form parameters. This depends on the used SPWare.
httpHeaders (DE/AT)	The HTTP headers append on the HTTP response message sent back to the user's browser

*Table 21 –Attributes of VIDPStartAuthResponse*

### 3.5.5 SPWareStartAuthRequest

This message is sent from the V-IDP to the SPWare to start an authentication process.

Attributes	Description
SPID (DE/AT)	This attribute uniquely identifies a service provider.
CCC (DE/AT)	The citizen country code to determine the SPWare to be used.
finalRedirectUrl (DE/AT)	The URL to which the browser should be redirected after the authentication process.
requestedAttributes (DE/AT)	The attributes which are requested by a SP.
SPCertSig (DE)	Certificate for the signature verification key of the SP. The SAML AuthnRequest is signed by the SP and the signature can be verified by using this certificate.
SPCertEnc (DE)	The attributes are returned encrypted. For encryption, the key of this certificate is used.
authLevel (DE/AT)	Requested authentication level.
SPAAuthRequest (DE)	If the VIDPStartAuthRequest is sent from a PEPS this attribute contains the original authentication request as sent from the SP. It is used to authenticate the SP to the eIdService.

*Table 22 – Attributes of SPWareStartAuthRequest*

### 3.5.6 SPWareGetAuthDataRequest

This message is sent from the V-IDP to one SPWare implementation to get the final authentication data.

Attributes	Description
sessionId	see VIDPGetAuthDataRequest.

*Table 23 – Attributes of SPWareGetAuthDataRequest*

### 3.5.7 SPWareGetAuthDataResponse

This message is sent from the SPWare to the V-IDP as answer to the SPWareGetAuthDataRequest and contains the authentication data.

Attributes	Description
SAMLAAssertion	see VIDPGetAuthDataResponse
status	see VIDPGetAuthDataResponse

*Table 24 – Attributes of SPWareGetAuthDataResponse*

### 3.5.8 SPWareStartAuthResponse

This message is sent from the SPWare to the V-IDP containing the SPWare specific redirect data.

Attributes	Description
httpStatusCode	see VIDPStartAuthResponse
content	see VIDPStartAuthResponse
httpHeaders	see VIDPStartAuthResponse

*Table 25 – Attributes of SPWareStartAuthResponse*