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

D5.8.3e Software Design for MW architecture for MW architecture

Deliverable Name :	**D5.8.3 Technical Design for PEPS and MW models**
Status :	**Final**
Dissemination Level :	**Public**
Due date of deliverable :	**May 31th 2011**
Actual submission date :	**November 11th 2011**
Work Package :	**WP5**
Organisation name of lead contractor for this deliverable :	**WP5**
Author(s):	**Ivo Sumelong, Armin Lunkeit, Bernd Zwattendorfer, Tim Schneider**
Partner(s) contributing :	**AT, DE**

Abstract: The document describes the components for the MW model. The focus is on the integration of the Austrian and German modules, its combination to a common interoperability layer and a modular architecture to extend the system (referred to as Modular Authentication Relay Service MARS).

Project co-funded by the European Community under the FP7 ICT Policy Support Programme

History

Version	*Date*	*Modification reason*	*Modified by*
0.1	17/01/2010	Initial draft	Ivo Sumelong
0.2		Review, Chapter regarding the German eID-Service architecture added	Armin Lunkeit
0.3		Review	Armin Lunkeit, Ivo Sumelong
0.4		Review and Submission to German STORK Consortium	Armin Lunkeit, Ivo Sumelong
0.5		Review and Corrections after comments from German STORK Consortium	Ivo Sumelong, Armin Lunkeit
0.6		Review Austria	Bernd Zwattendorfer
0.7		Review and Submission to German-Austria STORK Consortium	Ivo Sumelong Bernd Zwattendorfer
0.8	16/03/2010	Review and Corrections after comments from German-Austria STORK Consortium	Ivo Sumelong Bernd Zwattendorfer
0.9	31/03/2010	Review	Ivo Sumelong
0.10	June 2010	Editorial fixes and small updates	Tim Schneider
0.11	11/11/2011	Updates for V-IDP 2.0 and regarding reviews	Bernd Zwattendorfer, Tim Schneider
1.0	11/11/2011	Quality review	R.C. Wannee, A. van Overeem

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

Table of contents

List of figures

List of tables

List of abbreviations

<Abbreviation>	<Explanation>
MW	Middleware. Architecture of the integration of eIDs in Services, with a direct communication between SP and user's PC, without any central server. The term also refers to the piece of software of this architecture that executes on the user's PC.
C-PEPS	Citizen Country PEPS: PEPS in the citizen's origin country
VIDP	Virtual IDP. A system component helping to abstract Pan-European eID -interoperability. It either serves as a delegation component between the SP-MW or S-PEPS and the needed SPWare (appropriate MW server component) or enables an SP-MW to communicate with other C-PEPS.
MW	Middleware. Architecture of the integration of eIDs in Services, with a direct communication between SP and user's PC, without any central server. The term also refers to the piece of software of this architecture that executes on the user's PC.
PEPS	Pan European Proxy Service or Server
S-PEPS	Service Provider PEPS: PEPS in the Service Provider's country
SP	Service Provider
SPWare	Piece of software installed at the Service Provider, that complements the MW
UCA	User Centric Authentication, is an authentication use case whereby the User Agent acts as a gateway for all communication between SP MW
eID-Service	The German Middleware that authenticates German citizens during online authentication with the help of the Bürgerclient
AusweisApp/ Bürgerclient	Client middleware that runs on German citizens PC's
MOA-ID	Server-side middleware processing eID authentication in Austria
BKU	Client middleware to be used in Austria

Executive summary

This document is the software design for the MW architecture, especially of the MARS software architecture, a common building block used in STORK to achieve eID interoperability. The MARS software architecture (in opposition to the PEPS common code documented in D5.8.3c [5]) is used to map the STORK architecture to the German-Austrian Online Authentication infrastructure. The integration of that infrastructure into the STORK middleware has been shown by graphics and short explanations. It focuses on the technical components and their roles in the STORK middleware. Also low-level information, such as sequence diagrams etc. is provided.

The MARS software architecture was realized with the special requirements of countries which do not want or are allowed to have a central gateway for eID interoperability. So far, Germany and Austria are using this architecture to integrate their eID solutions into the interoperability framework but MARS was designed to be extensible. Beside the possibility to support more middleware solutions in the future, the MARS software can also be extended to realize a PEPS. While this has not been implemented so far, the Appendix gives some hints on how to do this.

This document is an annex of the D5.8.3 Technical Design for PEPS and MW models and interoperability document [2], in which you can find more details in the introductory chapters.

1 Introduction

1.1 Objective

This document presents the Software design of the components of the STORK MW architecture. It pretends to specify the behaviour of its components, in such a way that programmers can work with it.

The view which was offered by D5.8.3a, by business process, is now complemented with views by components and classes. Thus please note that this document is to be understood by programmers.

As this is one document of the deliverable D5.8.3 Technical Design, please refer to D5.8.3 for the other parts of the introduction.

1.2 Context

Germany and Austria decided the introduction of the electronic identification card which provides the possibility of an electronic identification and authentication of the eID card holder. This process requires a software component (client middleware “AusweisApp” or “BKU”) for the end-user and a server-side middleware called eID-Service (Germany) or MOA-ID (Austria) which is required to access the data stored electronically on the eID card.

The STORK approach requires the possibility of an electronic authentication and identification by the use of an infrastructure provided by the Member States.

This document presents information about the German eID approach using the AusweisApp and the eID-Service as well as the Austrian MOA-ID approach and their integration into the STORK architecture.

Moreover, this document handles different designs of User-Centric Authentication (UCA) whereby authentication requests and responses are directed through the user agent by using e.g. an S-PEPS as well as other various alternatives such as avoiding request/responses through the user agent by directly accessing the VIDP-WS interface. Moreover, there are scenarios whereby a SP will not like to expose the Authentication Service to end users and vice versa. After a successful authentication some SPs will prefer authentication status notifications with authentication data in one call (by S-PEPS) while others will prefer to pull the authentication data (VIDP-WS). In order to achieve this dynamism and flexibility, an optional configuration model architecture is required as well as a modular approach for integration of the respective components.

The above description can be summarized to four clear Use Cases defined in the STORK Architecture which include:

1. UC-AU-P-eIdService
2. UC-AU-M-eIdService
3. S-PEPS – VIDP – MOA-ID
4. SP-AT – VIDP – MOA-ID
5. SP – VIDP – C-PEPS

1.3 Glossary

The complete STORK glossary can be found on the STORK Website using the following link:

http://www.eid-STORK.eu/index.php?option=com_smf&Itemid=33&topic=42.0.

2 STORK Middleware

The STORK Middleware (or V-IDP from a functional point of view) acts as a gateway between SP, S-PEPS, SPWare and C-PEPS. To understand the role of the V-IDP in the general context, please refer to *Figure 2: System Context Diagram* of D5.8.3a.

2.1 General Architectural Approach

The middleware is based on the so-called MARS architecture. The plug-ons (e.g. V-SP/V-PEPS) handle requests from external systems (SP, User Agents, and PEPS) while plug-ins handle requests from the STORK Middleware connecting to external systems like the German eID-Service or the Austrian server-side middleware MOA-ID. *Figure 1* illustrates the various deployment options for the MARS architecture.

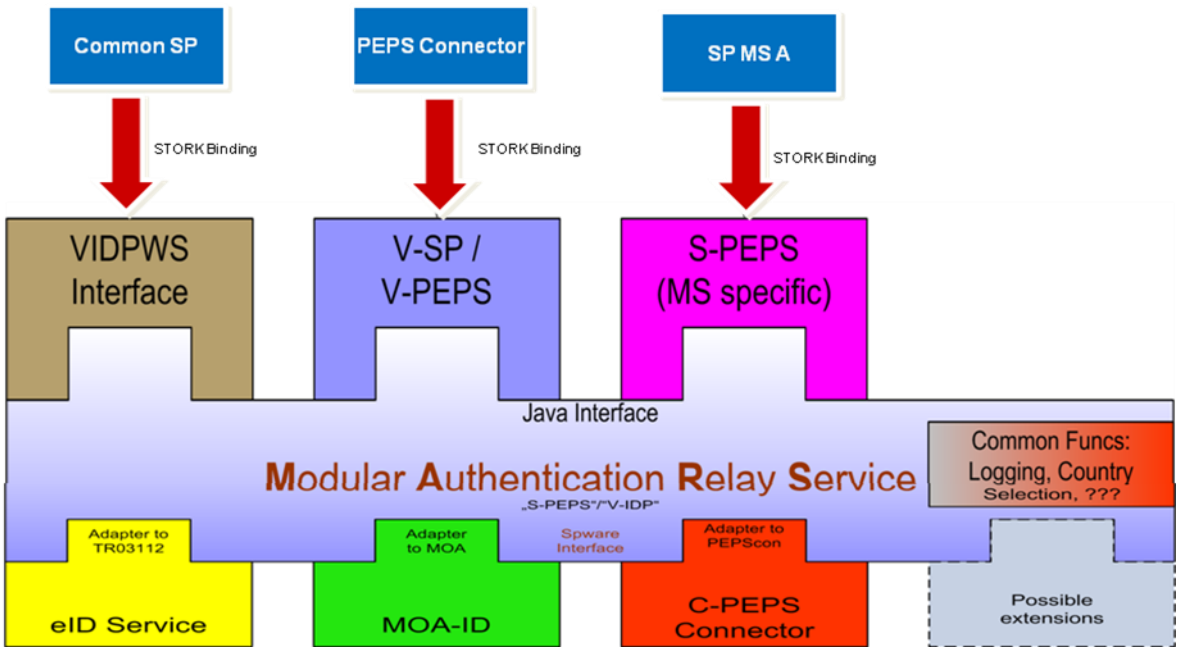


Figure 1: MARS architecture

2.2 Functional Requirement of the STORK Middleware

The MW will provide the following services to a SP, S-PEPS or C-PEPS.

- Handling all scenarios of a UCA
- Being able to route and receive calls from a C-PEPS
- Being able to route requests to the country-specific SPWare
- There are other functions of the VIDP such as certificate validation which are not part of the pilot project.

2.3 General STORK Authentication Reference Components

The diagram below presents components in the STORK authentication and their bindings.

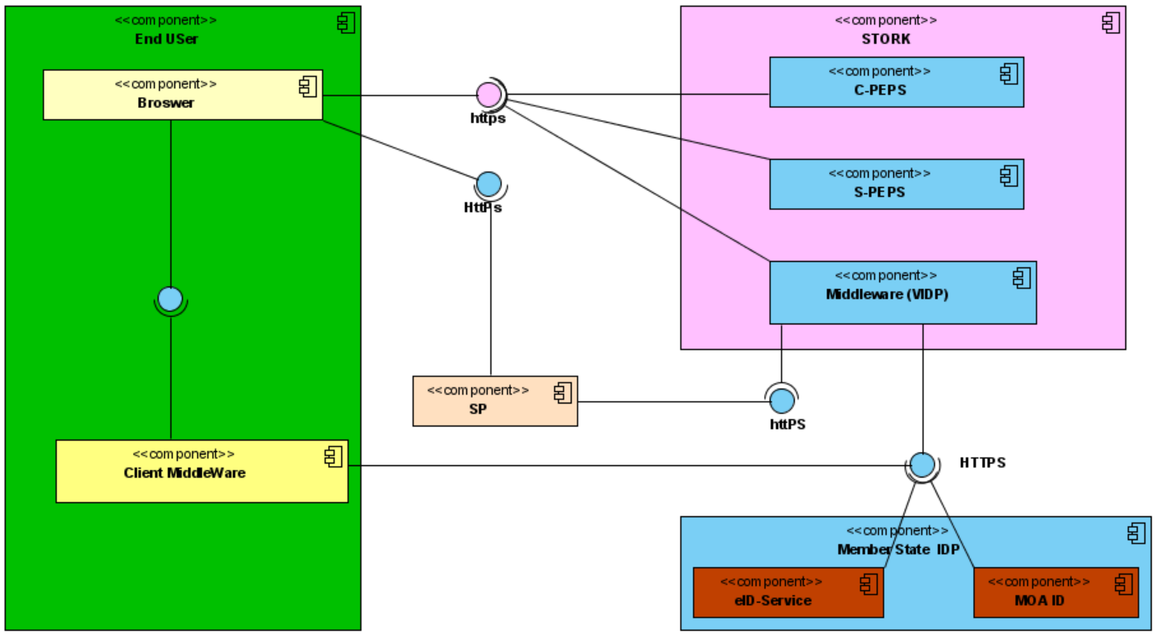


Figure 2: General STORK Authentication Reference Components

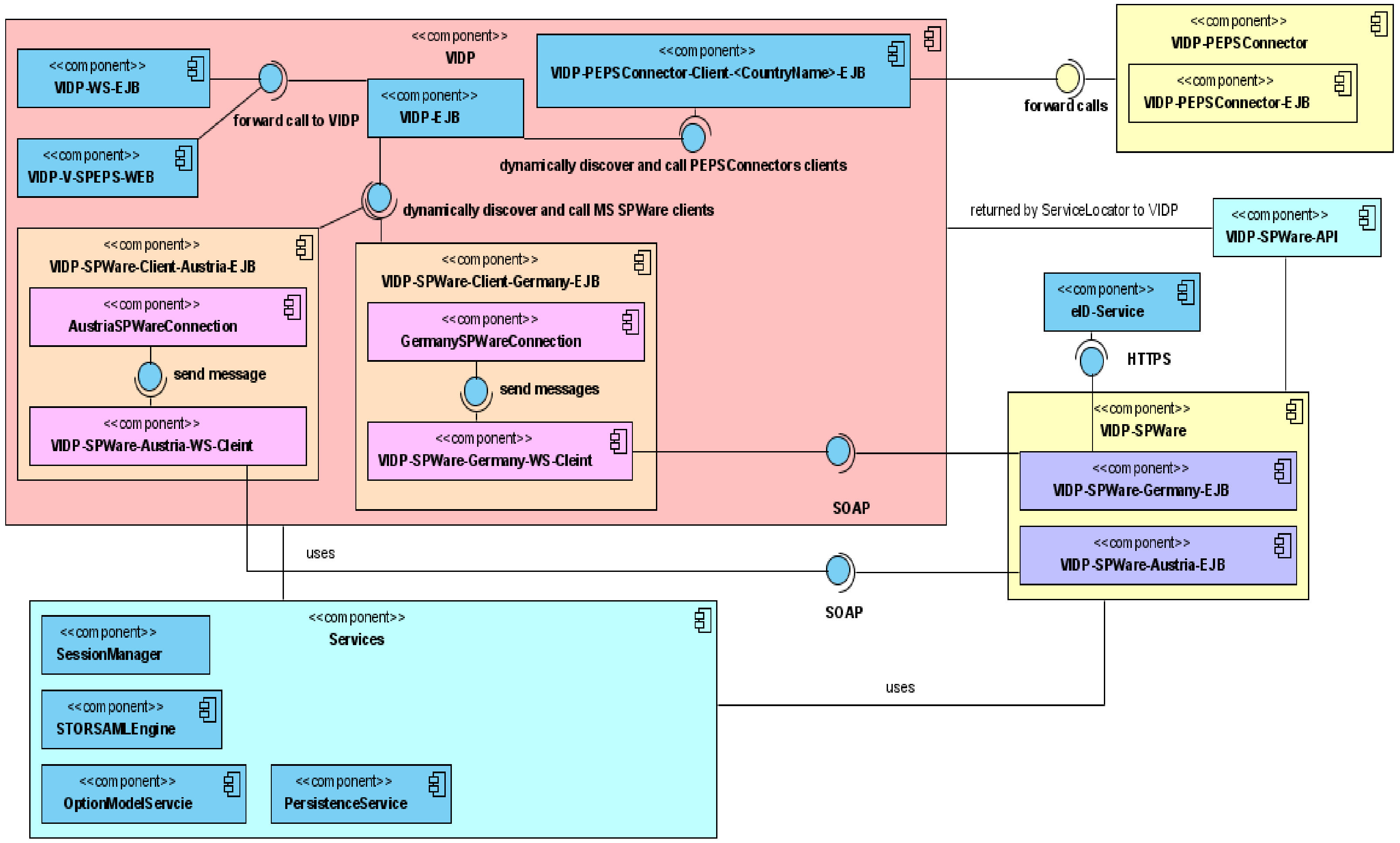
The following table provides a short description of the used components.

Reference Components of a General Authentication	
Name	Description
SP	The service provider requiring user authentication/identification
S-PEPS	Routes authentication request from SP to VIDP or other PEPS
VIDP	This handles the authentication process. It supports both scenarios of a UCA. It routs calls to C-PEPS or to SPWare like eID-Service
eID-Service	Handles in/outbound authentication messages as well as communication with the client middleware and encapsulates the German IDP
MOA-ID	Handles in/outbound authentication messages as well as communication with the client middleware and encapsulates the Austrian IDP
Client middleware	Authenticates and retrieves end user attributes from the electronic identity card
Browser	Intermediary between SP, VIDP, PEPS and client middleware.

Table 1: Reference Components of a General Authentication

2.4 Components of STORK MW

The figure below presents the various components of the STORK MW and their binding protocols followed by a table detailing their respective functions.



<<component>> VIDP
<<component>> VIDP-WS-EJB
<<component>> VIDP-V-SPEPS-WEB
forward call to VIDP
<<component>> VIDP-EJB
<<component>> VIDP-PEPSConnector-Client-<CountryName>-EJB
dynamically discover and call PEPSConnectors clients
dynamically discover and call MS SPWare clients
<<component>> VIDP-PEPSConnector
<<component>> VIDP-PEPSConnector-EJB
forward calls
returned by ServiceLocator to VIDP
<<component>> VIDP-SPWare-API
<<component>> VIDP-SPWare-Client-Austria-EJB
<<component>> AustriaSPWareConnection
send message
<<component>> VIDP-SPWare-Austria-WS-Cleint
<<component>> VIDP-SPWare-Client-Germany-EJB
<<component>> GermanySPWareConnection
send messages
<<component>> VIDP-SPWare-Germany-WS-Cleint
<<component>> eID-Service
HTTPS
<<component>> VIDP-SPWare
<<component>> VIDP-SPWare-Germany-EJB
<<component>> VIDP-SPWare-Austria-EJB
SOAP
uses
SOAP
<<component>> Services
<<component>> SessionManager
<<component>> STORSAMLEngine
<<component>> OptionModelServcie
<<component>> PersistenceService
uses

Figure 3: Component Diagram of STORK MW

Name	Description
VIDP-WS-EJB (web service)	Used to initiate authentication by SP when a secured resource is accessed by an end user Used to retrieve authentication information by SP
VIDP-V-PEPS-WEB	Receives calls from S-PEPS and forward them to VIDP as well as return calls to S-PEPS Used by S-PEPS to retrieve authentication data
VIDP-EJB	Receives calls from either VIDP-WS-EJB or VIDP-V-PEPS-WEB and forwards the message either to a PEPSConnector or SPWare Client using a service locator
VIDP-SPWare-Client-<CountryName>-EJB	creates and forward calls to MS specific SPWare implementation
VIDP-SPWare-API	Contains SPWare interfaces, exceptions, and messages as well as utility classes
VIDP-SPWare-<CountryName>-EJB	Contains MS specific SPWare implementation. Receives calls from VIDP, does attribute mapping if necessary and forwards calls to SPWare
VIDP-Services-EJB	Contain persistence, session, and option model services
OptionModelService	Used to read options for either SPWareConnector or PEPSConnector
SessionManager	Stores and retrieves all session information for authentication as well as context based optional model information
SAMLEngine	Creates and processes SAML Requests/Responses
VIDP-SPWareConnector-Client-<CountryName>-EJB	Provides connection services to SPWare service implementation. Acts as a client to SPWare implementation. This is the integration point of SPWare to VIDP
PersistenceService	Provides CRUD services for all persistence tables like those described in chapter 7
LogService	Logs all in/outbound messages

Table 2: Components of STORK Middleware

The components named VIDP-SPWareXXX<CountryName>-EJB are country-specific components which are currently implemented by Austria and Germany. Future plug-ins need to implement new instances of those components.

All other components are universal to the V-IDP. Please note that the VIDP-V-PEPS-WEB (plug-on) is only supposed to be used to communicate with S-PEPSes and that VIPD-WS-EJB web service is currently only used by German SPs.

There might be more plug-ons in the future depending on the needs of one country to let SPs directly communicate with the V-IDP but the web service plug-on is meant to be universal and is not restricted for use with Germany alone.

.

3 Middleware Components Interfaces

This chapter presents the interfaces and messages of the various components of the MW

3.1 VIDP

The VIDP handles the routing of calls from plug-ons to SPWare or C-PEPS (using PEPSConnector).

Methods	Description	Comments
startAuthentication	Called by plug-on (e.g. VIDP-WS or V-SP/V-PEPS) to initiate an authentication	
getAuthenticationData	Called by a plug-on to retrieve authentication data from a Member State MW (e.g. eID-Service)	

Table 3: VID Description of VIDP Functions

Message	Description	Type and Default
Inbound		
STORKAuthnRequest	STORK SAML Request Object. See details below	STORKAuthnRequest
Outbound		
StartAuthResponse	Not nullable,	StartAuthResponse
Exception	VIDPException, thrown when exception occurs within the VIDP or SPWare	

Table 4: StartAuthentication Method

getAuthenticationData		
Message	Description	Type and Default
Inbound		
VIDPGetAuthDataRequest	Holds session and country information	VIDPGetAuthDataRequest
Outbound		
STORKResponse	Holds SAMLResponse information, Not nullable,	STORKResponse
Exception	VIDPException, when exception occurred within, VIDP or returned by SPWare	

Table 5: GetAuthenticationData Method

3.2 SPWare

It acts as an Integrator between the VIDP and member states SPWare (e.g. eID-Service). It handles – if required – translations from and to STORK attributes to respective member state attributes and STORK error codes mappings.

Methods	Description	Comments
startAuthentication	Called by VIDP to initiate an authentication at member state MW (e.g. eID-Service)	
getAuthenticationData	Called by VIDP to retrieve authentication data from a member state MW (e.g. eID-Service)	

Table 6: Description of SPWare Methods

Message	Description	Type and Default
Inbound		

STORKAuthnRequest	STORK SAML Request Object. See details below	STORKAuthnRequest
Outbound		
StartAuthResponse	Not nullable,	StartAuthResponse
Exception	SPWareException, thrown when an exception occurs within the SPWare	

Table 7: StartAuthentication Method

Message	Description	Type and Default
Inbound		
GetAuthDataRequest	Holds session information	GetAuthDataRequest
Outbound		
STORKResponse	Holds SAMLResponse information, Not nullable,	STORKResponse
Exception	SPWareException, thrown when an exception occurred in a SPWare implementation or an error code is returned by SPWare or C-PEPS	

Table 8: GetAuthenticationData Method

3.3 Dynamic integration of plug-on and plug-ins.

Plug-in Modules

These modules could be either web or ejb.

Implementation

1. Create ejb module as a MS SPWare
2. Download ***VIDP-spware-api-1.0-RELEASE.jar*** and ***mw-persistence-api-1.0-RELASE.jar*** from [***https://vidp.openlimit.com:8120/artifactory***](https://vidp.openlimit.com:8120/artifactory) and create dependencies to them
3. Create remote and local interfaces that extend eu.stork.vidp.spware.api.interfaces.SPWare interface
4. Implement both interfaces. In the implementation by injection, reference PersistenceManager and SessionManager as follows:

    @EJB(mappedName = "SessionManagerBean")

    private SessionManagerRemote sessionManagerBean;

    @EJB(mappedName = "PersistenceServiceBean")

    private PersistenceServiceRemote persistenceServiceBean;

    - Always ensure to validate the session state based on the transaction function call.
    - Any failed transaction should lead to an update of the session with a mapping of the SPware error to STORKError code. A corresponding *SPWareException* must be returned
    - See STORK spec on how to handle attributes requested under (unsupported|mandatory and unsupported|optional)

5. Deploy the ejb module or package it within the VIDP.ear Really within here or as part of VIDP-SPware.ear

Configurations:

1. Ensure the SPWares country is configured in ol_country (a list of all countries)and included in the list of *ol_storkmemberstates*
2. Configure the *SPWare* in *stork_live* database within the *ol_ SPware* table. The countryID of the SPWare is mandatory.

3. Configure a contextframe for the SPWare in *ol_contextframe*. The entry should contain only the SPWareID and other mandatory entries.
4. Go to *ol_optionItem* and configure the SPWare as follows
   - Find the id of the *isSPWareRemoteInterfaceEnabled* in ol_option
   - Create an entry in *ol_optionItem* using the optionID of the *isSPWareRemoteInterfaceEnabled* and the contextframeID from *ol_contextframe*, set the BooleanValue column to true
   - Find the id of an optionName "Remote-API" under optiongroupName="SPWareAPIs" in *ol_option*
   - Create an entry in *ol_optionItem* using the optionID of the Remote-API and the contextframeID from 3 above, set the StringValue=canonical name of the SPWare remote interface
5. Do SPWare Error Code –STORK Error Code Mapping within *ol_spwareerrorcode*
6. Do SPWare-STORK Attribute mapping in *ol_countryattribute*

Plug-ons

- Create a web or ejb module with reference to VIDP-Client-API.
- Use the VIDPClient to forward all calls to VIDP.

In the implementation by injection reference PersistenceManager and SessionManager as follows:

```
@EJB(mappedName = "SessionManagerBean")
private SessionManagerRemote sessionManagerBean;
@EJB(mappedName = "PersistenceServiceBean")
private PersistenceServiceRemote persistenceServiceBean;
```

- Always ensure to create the session before calling *startAuthentication* and delete it after *getAuthenticationData*.
- Any failed transaction should lead to an update of the session with the STORKErrorcode
- Ensure to initialise the STORKSAMLEngine only once.

3.4 VIDP-WS-DE (The German Web Service and Interfaces)

The VIDP-WS (web service) is described in this section. The internal implementation of this service is not mentioned here. Described issues include methods, in/outbound messages, and codes (status, error and attribute status). For every message there is a sample provided to give a clearer view of any message description.

Methods	Description	Comments
initAuthentication	Called by SP to initiate an authentication	
getAuthenicationData	Called by SP to retrieve authentication data	
isLive	In Live production to avoid calling business methods that could lead to financial cost and long execution time, the isLive method is used to check if service is responding to requests without actually calling initAuthentication or getAuthenicationData	Always return success if service is running.

Table 9: VIDP-WS General Description of Methods

3.4.1 initAuthentication

Message	Description	Type and Default
Inbound		
RequestControl	Is a MesssageControlType. See Table 11.	MesssageControlType
SPControl	Holds information for protection of services as well as VIDP control.	SPControl See structure in sample below
PersonConfig	Holds attribute information required by SP. For message details see sample authenticationRequest below as well as the special table titled "PersonConfig"	
Outbound		
Response	Not nullable	See section 3.1.2
Exception	No exception. Even SAOPFault exception are avoided	

Table 10: initAuthentication

Attribute	Description	Type and Default
TransactionID	Identifier for this conversation	String, not null
CreateTimestamp	The time the request was created	Timestamp, not null
ValidStartTimestamp	The time the request starts to become valid	Timestamp, not null
ValidEndTimestamp	The time the request becomes invalid	Timestamp, not null
Comment	*Conversation=initAuthentication+getAuthenticationData*	

Table 11: The MesssageControlType

Attribute	Description	Type and Default
SPID	Unique Identifier of an SP obtained from the	String, not null
Domain	Domain of the SP	URI, not null
Application	Application found within the domain of the SP	URI, not null
QAALevel	The QAALevel	int, not null
CountryCode	The citizen's member state ISO name	String, not null
ServiceProviderIssuerURL	SP inique URL identifier	URI, not null
AssertionConsumerService URL	URL to return authentication responses	URI, not null
Comments		

Table 12: SPControl

Sample request by initAuthentication

```
<AuthenticationRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="STORK-VIDP-WS.xsd">
<!—Element RequestControl and its child elements are mandatory-->
 <RequestControl>

            <TransactionID>Trans 2002-10-10T12:</TransactionID>
            <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
            <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
            <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
</ReqesutControl>
<!—Element SPControl and its child elements are mandatory-->

<SPControl>
            <SPID>SP  Muster ID XXXXX</SPID>
             <CountryCode> User Member States ISO Name(AT)</Country>
             <Domain> SP Muster http:www.sp.de </ Domain >
             <Application> http:www.sp.de/Shop</Application>
            <QAALevel> SP Muster QAA </QAALevel>
            < AssertionConsumerURL > SP Muster http:www.sp.de</ AssertionConsumerURL >
            < ServiceProviderIssuerURL > SP Muster http:www.sp.de</ ServiceProviderIssuerURL >
</SPControl>

<!—Element PersonConfig is mandatory-->

<PersonConfig>

            <!--Element DocumentType  is optional-->
            <DocumentType Required="true"/ >
            <!--Element IssuingState is optional-->
            <IssuingState Required="true"/ >
             <!--Element DateOfExpiry is optional-->
            <DateOfExpiry Required="true" />
            <!--Element GivenNames is optional-->
            <GivenNames Required="true" />
            <!--Element FamilyName is optional-->
            <FamilyName Required="true"/ >
            <!--Element ArtisticName is optional-->
            <ArtisticName Required="true" />
             <!--Element AcademicTitle is optional-->
            <AcademicTitle Required="true" />
            <!--Element DateOfBirth is optional-->
            <DateOfBirth Required="true" />
             <!--Element PlaceOfBirth is optional-->
            <PlaceOfBirth Required="true" />
            <!--Element PlaceOfResidence is optional-->
            <PlaceOfResidence Required="true" />
            <!—Element RestrictedIdentification is optional-->
            <RestrictedIdentification Required="true" / >
             <!--Element CommunityIDVerification is optional-->
            <CommunityIDVerification Required="true"  Criteria="string"/>
            <!--Element AgeVerifcation is optional-->
            <AgeVerifcation Required="true"  Criteria="string"/>
            <!--Element DocumentValidity is optional-->
             <DocumentValidity Required="true"  Criteria="string"/>
             <!--Element Age is optional-->
            <Age Required="true" />
            <!--Element Gender is optional-->
            <Gender Required="true" />
            <!--Element NationalityCode is optional-->
            <NationalityCode Required="true" />
            <!--Element Marital Status  is optional-->
            <MaritalStatus Required="true" />
            <!--Element ResidencePermit  is optional-->
            <ResidencePermit Required="true" />
            <!--Element TextResidenceAddress  is optional-->
            <TextResidentAddress Required="true" />
```

Sample request by initAuthentication

```
            <!--Element Email  is optional-->
            <Email  Required="true" />
</PersonConfig>
</AuthenticationRequest>
```

Table 13: Sample request by initAuthentication

3.4.2 Response

This object is returned during any conversation with the service. Its states depend on the type of conversation and possible processing status as well as errors. It carries personal data and authentication data in case a **`getAuthenticationData`** call was made by the SP.

Parameter	Description	Type and Default
RequestControl	The RequestControl sent by SP is returned	MessageControlType, not null
ResponseControl	A new MessageType from Middleware to SP. Its transactionID should be same as that in RequestControl, not null	MessageControlType, not null
Status	int	0 for SUCCESS else 1 for FAILURE, 2 for PENDING (Currently Status=2 is not supported)
HttpObject	String, nullable	Not null when status is 0 else null and set only during initAuthentication. Its content is closed in a character data and should be redirected to the User-Agent
Person	nullable	Set only during getAuthenticationData and when status is 0. It holds the authentication data returned from VIDP. See Authentication Data example below.
Error	Error, nullable	Not null when status is 1 else null. Contains error code and message. Set in case of an error during initAuthentication and getAuthenticationData. See section 3.1.3

Parameter	*Type*	*Default*
ErrorCode	Int	
Message	String, not null	

Table 14: Description of Response Message

Sample Response by initAuthentication with success

```
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="STORK-VIDP-WS.xsd">

<!--Element RequestControl  is mandatory-->
 <RequestControl>
              <TransactionID>Trans 2002-10-10T12:</TransactionID>
              <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
              <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
              <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
</ReqesutControl>
<!--Element ResponseControl  is mandatory-->
 <ResponseControl>
              <TransactionID>Trans 2002-10-10T12:</TransactionID>
              <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
              <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
              <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
</ResponseControl>
<Status>0</Status>
<HttpObject>
<![CDATA[<HTML><HEAD>
   <TITLE> eCard Client Initiator
   </TITLE> </HEAD>   <BODY>
      <object type="application/vnd.ecard-client">
     <param name="ServerAddress" value="eID-Service IP"/>
     <param name="SessionIdentifier" value="123456"/>
     <param name="Binding" value="urn:liberty:paos:2003-08"/>
     <param name="PathSecurity-Protocol" value="uri:iso:PAOS"/>
     <param name="PathSecurity-Parameters"
     value="12345678912345678987654321987654321123456789123456789876543212345678987654321234567
898765432123987654321234567898765432123456789876543212345678987654321234567898765432123456789
8765432112345678912345678987654321234567898765432123456789876543212345678987654321234567898765
5FF"/>      <param name="RefreshAddress" value="https://www.SP.de/AssertionConsumerURL"/>
   </object>  </BODY></HTML>]]>

</HttpObject>
</Response>
```

Table 15: Sample Response by initAuthentication with success

Sample response by initAuthentication with failure

```
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="SPAuthenticationService.xsd">
<!--Element RequestControl  is mandatory-->
 <RequestControl>
              <TransactionID>Trans 2002-10-10T12:</TransactionID>
              <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
              <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
              <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
</ReqesutControl>
<!--Element ResponseControl  is mandatory-->
 <ResponseControl>
              <TransactionID>Trans 2002-10-10T12:</TransactionID>
              <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
              <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
              <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
</ResponseControl>
<Status>0</Status>
 <Error><Message>VIDP unknown internal error </Message>
      <ErrorCode>100</ErrorCode> </Error> </Response>
```

Table 16: Sample Response by initAuthentication with failure

3.4.3 getAuthenticationData

Message	Description	Type and Default
Inbound		
Timestamp	Timestamp during request	Timestamp, not null
TransactionID	Identifier for this conversation	String, not null
PolicyInfo	Policy used by client	Not null
SPAuthentication	Holds information for authentication and authorization and process control at VIDP	Not Nullable
MessageID	Identifier for this request	String, not null
Outbound		
Response	Holds the authentication data (Person object), not null	Response (see description in the table below)
Exception	No Exception is thrown, also SOAPFault are avoided	

Table 17: Description of getAuthenticationData Method

Sample request by getAuthenticationData

```
GetAuthenticationDataRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="STORK-VIDP-WS.xsd">
<!—Element RequestControl and its child elements are mandatory-->
<RequestControl>

            <TransactionID>Trans 2002-10-10T12:</TransactionID>
            <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
            <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
            <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
</ReqesutControl>
<!—Element SPControl and its child elements are mandatory-->

<SPControl>
            <SPID>SP  Muster ID XXXXX</SPID>
             <CountryCode> User Member States ISO Name(AT)</Country>
             <Domain> SP Muster http:www.sp.de </ Domain >
             <Application> http:www.sp.de/Shop</Application>
            <QAALevel> SP Muster QAA </QAALevel>
            < AssertionConsumerURL > SP Muster http:www.sp.de</ AssertionConsumerURL >
            < ServiceProviderIssuerURL > SP Muster http:www.sp.de</ ServiceProviderIssuerURL >
</SPControl>

GetAuthenticationDataRequest>
```

Table 18: Sample Request by getAuthenticationData

Sample response by getAuthenticationData with success status

```
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="
STORK-VIDP-WS.xsd">
<!--Element RequestControl  is mandatory-->
 <RequestControl>
            <TransactionID>Trans 2002-10-10T12:</TransactionID>
            <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
            <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
            <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
</ReqesutControl>
<!--Element ResponseControl  is mandatory-->
 <ResponseControl>
            <TransactionID>Trans 2002-10-10T12:</TransactionID>
            <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
            <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
            <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
```

```
</ResponseControl>
<Status>0</Status>
<!--AuthenticationData is optional and appears only when getAuthenticationData and when status above is 0-->
<Person>
            <!--Element DocumentType is optional-->
            <DocumentType>string</DocumentType>
            <!--Element IssuingState is optional-->
            <IssuingState>string</IssuingState>
            <!--Element DateOfExpiry is optional-->
            <DateOfExpiry>1999-01-21</DateOfExpiry>
            <!--Element GivenNames is optional-->
            <GivenNames>string</GivenNames>
            <!--Element FamilyName is optional-->
            <FamilyName>string</FamilyName>
            <!--Element ArtisticName is optional-->
            <ArtisticName>string</ArtisticName>
            <!--Element AcademicTitle is optional-->
            <AcademicTitle>string</AcademicTitle>
            <!--Element DateOfBirth is optional-->
            <DateOfBirth>1999-01-21</DateOfBirth>
            <!--Element PlaceOfBirth is optional-->
<PlaceOfBirth>
            <StreetName>string</StreetName>
            <StreetNumber>string</StreetNumber>
            <AppartmentNumber>string</AppartmentNumber>
            <PostalCode>string</ PostalCoder>
            <Town>string</Town>
            <Municipality>string</ Municipality >
            <State>string</State>
            <CountryCode>string</CountryCode>
</PlaceOfBirth>
<!--Element PlaceOfResidence is optional-->
 <PlaceOfResidence>
            <StreetName>string</StreetName>
            <StreetNumber>string</StreetNumber>
            <AppartmentNumber>string</AppartmentNumber>
            <PostalCode>string</ PostalCoder>
            <Town>string</Town>
            <Municipality>string</ Municipality >
            <State>string</State>
            <CountryCode>string</CountryCode>
</PlaceOfResidence>
<!--Element RestrictedId is optional-->
<RestrictedIdentification>1234567890ABCDEF</RestrictedIdentification>
<!--Element CommunityIDVerification is optional-->
<CommunityIDVerification>
            <Request>string</Request>
            <Result>true</Result>
</CommunityIDVerification>
<!--Element AgeVerifcation is optional—
<AgeVerifcation>
            <Request>65535</Request>
            <Result>true</Result>
            </AgeVerifcation>
<!--Element DocumentValidity is optional-->
<DocumentValidity>
            <ReferenceDate>1999-01-21</ReferenceDate>
            <Status>string</Status>
</DocumentValidity>
</Person>
</Response>
```

Table 19: Sample Response by getAuthenticationData with success status

Sample response by getAuthenticationData with failed status
(see below)

```
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="
STORK-VIDP-WS.xsd">
<!--Element RequestControl  is mandatory-->
 <RequestControl>
            <TransactionID>Trans 2002-10-10T12:</TransactionID>
            <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
            <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
            <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
</ReqesutControl>
<!--Element ResponseControl  is mandatory-->
 <ResponseControl>
            <TransactionID>Trans 2002-10-10T12:</TransactionID>
            <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
            <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
            <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
</ResponseControl>
<Status>1</Status>
<Error>
            <Message>VIDP unknown internal error </Message>
            <ErrorCode>100</ErrorCode>
</Error>
</Response>
```

Table 20: Sample response by getAuthenticationData with failed status

3.4.4 isLive

Used by SP to find out service availability thus avoiding financial cost or performance degrading in live production.

Message	Type and Default	
Inbound		
PingRequest	<table><tr><th>Parameter</th><th>Type, Description</th></tr><tr><td>RequestControl</td><td>MessageControlType</td></tr><tr><td>SPControl</td><td>SPControl</td></tr></table>	
Outbound		
Response	Holds error if an error occurred else carries only status.	Response (see description in the table below)
Exception	No Exception is thrown, also SOAPFault are avoided	

Table 21: Description of isLive Method

Sample isLive request

```
PingRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="
STORK-VIDP-WS.xsd">
<!—Element RequestControl and its child elements are mandatory-->
 <RequestControl>

            <TransactionID>Trans 2002-10-10T12:</TransactionID>
            <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
            <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
            <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
</ReqesutControl>
<!—Element SPControl and its child elements are mandatory-->

<SPControl>
            <SPID>SP  Muster ID XXXXX</SPID>
             <CountryCode> User Member States ISO Name(AT)</Country>
             <Domain> SP Muster http:www.sp.de </ Domain >
             <Application> http:www.sp.de/Shop</Application>
            <QAALevel> SP Muster QAA </QAALevel>
            < AssertionConsumerURL > SP Muster http:www.sp.de</ AssertionConsumerURL >
            < ServiceProviderIssuerURL > SP Muster http:www.sp.de</ ServiceProviderIssuerURL >
</SPControl>

</PingRequest>
```

Table 22: Sample isLive request

Sample isLive response

```
Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="
STORK-VIDP-WS.xsd ">
   <!--Element RequestControl  is mandatory-->
 <RequestControl>
            <TransactionID>Trans 2002-10-10T12:</TransactionID>
            <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
            <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
            <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
</ReqesutControl>
<!--Element ResponseControl  is mandatory-->
 <ResponseControl>
            <TransactionID>Trans 2002-10-10T12:</TransactionID>
            <CreateTimeStamp>2002-10-10T12:00:00-05:00</CreateTimestamp>
            <ValidStartTimeStamp>2002-10-10T12:00:00-05:00</ ValidStartTimeStamp >
            <ValidEndTimeStamp>2002-10-10T12:00:00-05:00</ ValidEndTimeStamp >
</ResponseControl>
   <Status>0</Status>
</PingResponse>
```

Table 23: Sample isLive Response

3.5 SP-MW Adapter AT (Web)

The Austrian SP-MW Adapter offers service providers a web interface to start an authentication request by simply using URL parameters. Additionally, this adapter offers a template for country selection. The authentication process is started by transferring the URL request into a SAML AuthnRequest and sending the request to the VIDP. The actual user authentication is still done by the Austrian middleware MOA-ID. The adapter is only invoked again when the SAML authentication data is ready to be retrieved. After successful authentication a MOA-ID assertion is constructed. For this assertion a SAML artifact is generated and transmitted to the calling SP for assertion retrieval. This adapter enables Austrian legacy applications to provide a secure authentication of foreign citizens.

An authentication process can be started by simply calling a URL. Such a URL should consist of the following parameters:

URL parameter (mandatory)

Name	Range	Description
SPID	String	ID of the Service Provider
OA	anyURI	URL of the SP to which the user should be redirected after successful authentication

Table 24: Mandatory parameters in the URL for the AT SP MW adaptor

URL parameter (optional)

Name	Range	Description
QAALevel	1 to 4	Desired authentication level
CCC	Two-digit ISO country code	Country code of a citizen's home country. If not provided, a country selection page will be displayed.
AttributeList	Format: attrName,isRequired,value;	Additional attributes to eID, first name, last name, date of birth, nationality code according to section 7.2 of deliverable D.5.8b (only suffix) isRequired: true or false value: base value Separation between attributes with ";"
Target	String	Sector, only required for AT authentication
Template	anyURI	Optional HTML template for MOA-ID authentication
bkuURI	anyURI	URL of the client middleware to be used

Table 25: Optional parameters in the URL for the AT SP MW adaptor

Sample URL:

```
http://localhost:8080/moa.stork.web/STORKStartAuthentication?
SPID=SP-AT&OA=http://localhost:8082/moa-id-proxy/&
Target=Test&QAALevel=4&AttributeList=age,false;isAgeOver,false,16
```

If no citizen country is provided, a country selection page will be displayed.

The optional parameters "QAALevel" and "AttributeList" can also be specified in the configuration of this module.

3.6 V-PEPS/V-SP (Web)

The V-PEPS module receives requests from an S-PEPS and forwards it to the VIDP which in turn determines the SPWare to invoke. Additionally, information received from the VIDP must be converted to browser responses.

3.6.1 Process Flow

The V-PEPS module has to cope with two different situations which are described in detail below. Situation 1 describes the processing after an S-PEPS has forwarded a SAML AuthnRequest (which needs to be compliant to the message format defined in D5.8.3b [4]) to the VIDP for authenticating a MW citizen. The validity of the SAML AuthnRequest must be verified and the appropriate national module needs to be called. Situation 2 describes the processes after being returned from the national MW module (currently the Austrian and German SPWare). In that situation the V-PEPS module calls the VIDP to retrieve citizen's authentication data and returns them to the requesting S-PEPS.

Situation 1:

- Receipt of SAML AuthnRequest via HTTP Post
- Decoding of SAML AuthnRequest
- Validation of SAML AuthnRequest
    - Validate format (XML syntax)
    - Validate digital signature
    - Validate format of SAML AuthnRequest
    - Validate contents of SAML AuthnRequest elements and attributes
- Calling VIDP with SAML AuthnRequest
- Return browser response received from VIDP

Situation 2 (return from German or Austrian SPWare, not PEPSConnector):

- Receipt of HTTP GET Request with parameters sessionID and citizen country code
- Calling VIDP with parameters
- Return SAML response as HTTP Post to the user's browser

3.7 PEPSConnector

The so-called PEPSConnector is a special implementation of the SPWare interface and is responsible for handling authentication request and response messages between a VIDP and a C-PEPS. The PEPSConnector itself can be seen as two layer architecture, one layer handling web requests and thus building the front-end layer whereas the back-end layer is responsible for handling the business logic.

3.7.1 Back-End

The back-end of the PEPSConnector module implements the methods of the SPWare Interface (see Table 26 for details).

Methods	Description	Comments
startAuthentication	Called by the VIDP to send an authentication request to the appropriate C-PEPS	
getAuthenticationData	Called by the VIDP to retrieve authentication data from a C-PEPS	

Table 26: Description of PEPSConnector Methods

3.7.1.1 Process Flow

3.7.1.1.1 startAuthentication

The following actions of the PEPSConnector must be carried out within this method:

- Receive SAML2 AuthnRequest object from VIDP
- Locate foreign C-PEPS and fill in missing values in the request (destination, etc.)

- Get signing credentials and sign the request
- Include request in HTML page according SAML specification
- Store sessionID in connection with AuthnRequest
- Return HTML page as byte stream

3.7.1.1.2 getAuthenticationData

This method is responsible for processing the following steps:

- Get sessionID from VIDP
- Fetch response
- Return SAML response

3.7.2 Front-End

The PEPSConnector Front-End defines the web interface to the C-PEPS for receiving appropriate SAML response messages.

3.7.2.1 Process Flow

The following actions must be performed after having received a SAML response message from a C-PEPS:

- Validate response and assertion
- Store response in connection with session
- Redirect user to the actual calling module (plug-on) given the URL in the session

4 Application, Modules and Packaging

To allow for more flexibility in deployment and maintenance this chapter describes the deployments and packaging approach. It provides the deployment strategies, various applications, modules and packaging. Some major considerations are:

- Options to integrate a SPWare into a VIDP at runtime
- The integration layer should have minimal effort in implementation. In best case code generation tools can handled since no business logic is included in this artefact.
- More effort with configuration than coding
- Completely decouple modules from each other through the use of modules termed APIs giving room for a dynamic approach in deployment. A module can be moved from one application to another or from one system to another without breaking down the application functionality at runtime. Each module should be able to handle local, remote (EJB) and web service integration using just one defined interface (POJO-local).
- Each SPWare client is encapsulated in an EJB module containing a stateless session to address performance issues as well. The VIDP can maintain specific number of open connections to a specific SPWare by using the non-transaction, stateless session beans. During their creation, web service port instance of a SPWare client is created. They also handle issues like transaction timeouts.

4.1 Applications

These are J2EE applications that can be deployed[1] on same virtual machine, same application server or different virtual machines.

The figures below show different deployment options. Due to the modular approach other deployment options are possible. Such a deployment option could be deployment on one single application server. It shows dependencies within artefacts in the application as well as direct application dependencies. Please note that the SPWares always have to be in a separate application, but can be deployed on the same application server.

[1] Note that due to missing serializing functionality of OpenSAML library, a distributed deployment across different application servers or even machines is currently not possible. This means that for the time being the V-IDP needs to be installed completely inside one application server.

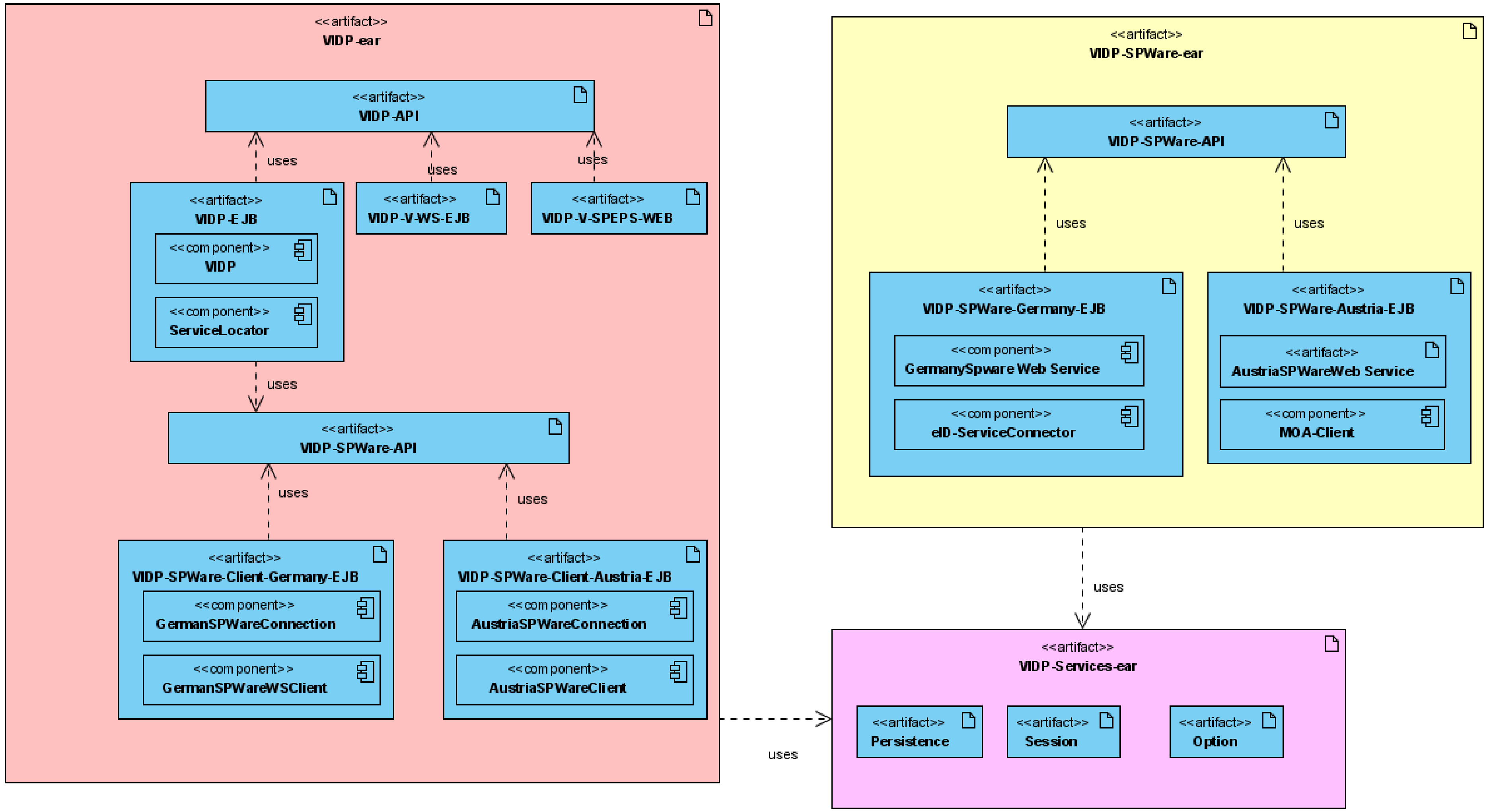


Figure 4: Architecture of STORK Middleware. Deployment on single application server as three applications.

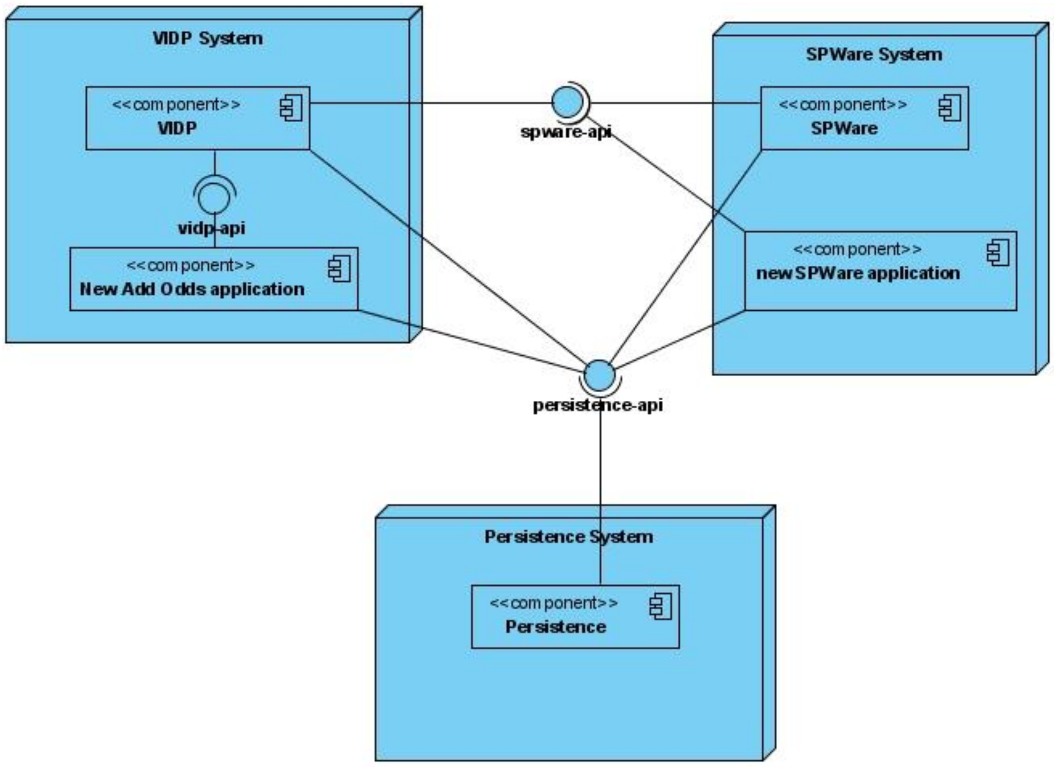


Figure 5: Deployment of applications on different systems. New Plug-ons/in package and deployed as new applications

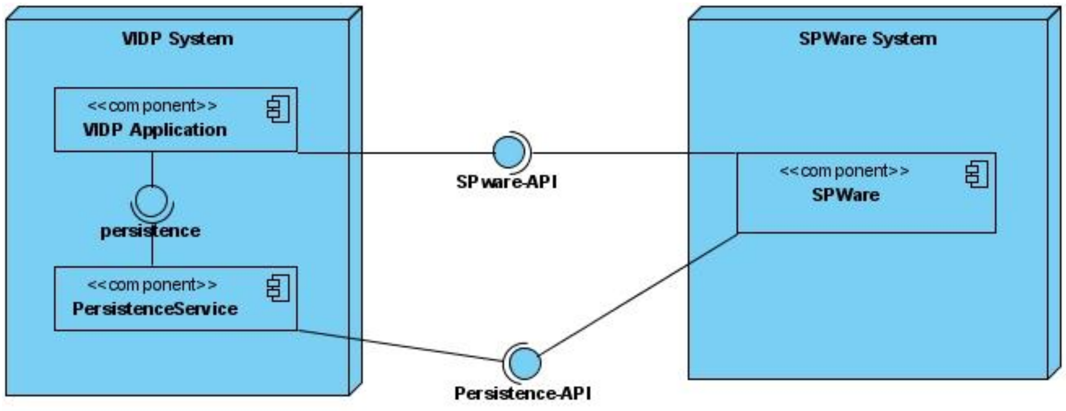


Figure 6: Deployment of applications on different systems (persistenceService together with VIDP). New Plug-ons/in package within existing VIDP and SPWare

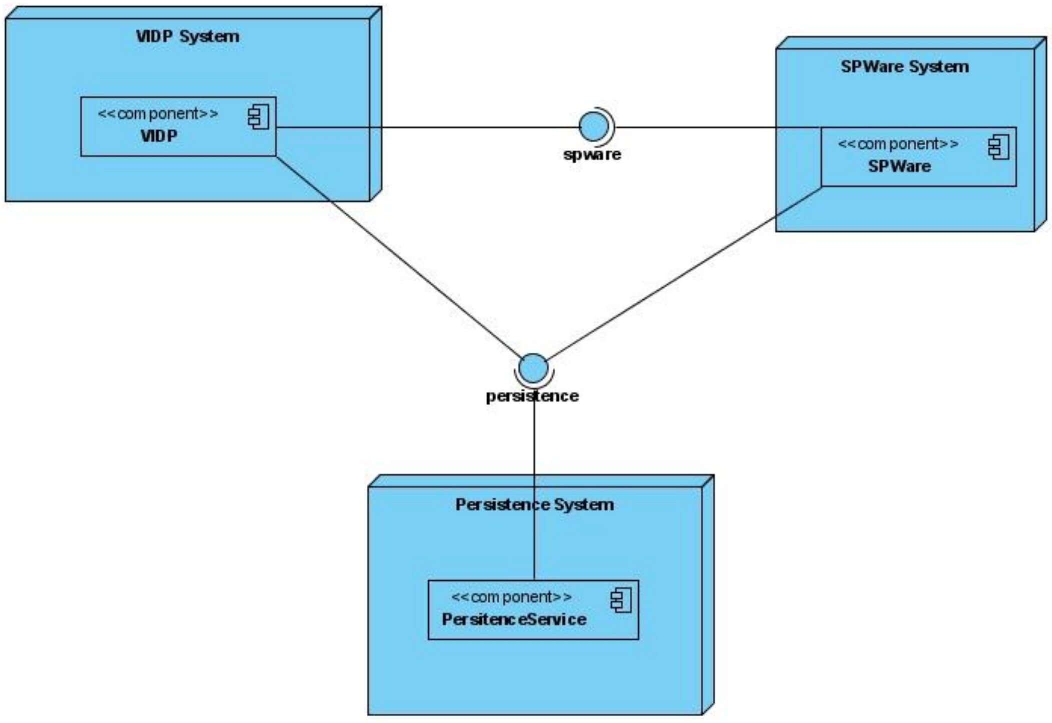


Figure 7: Deployment of applications on different systems. New Plug-ons/in package within existing VIDP and SPWare

Name	Description	Comments
VIDP	contains modules for VIDP implementation, VIDP-SPEPS, VIDP-WS and SPWare clients as well as PEPSConnector	
SPWare	Contains implementation of SPWare web service for each Member State	
VIDPServices	Contains remote services accessible by both VIDP and SPWare such as SessionManager, OptionModelService, and PersistenceServices	
MWSecurityGateway	An application service that will handle security issues like transport and message security, policy enforcement point, validations, authorization to framework and ...)	
Comments		

Table 27: Sample Applications

4.2 Modules

Name	Description
VIDP-EJB	Contain local, stateless implementation of VIDP as well as the service locator for dynamically discovering and calling SPWare clients and PEPSConnectors
VIDP-WS-EJB	Contains the enterprise web service implementation of VIDP-WS
VIDP-V-SPEPS-WEB	Contains implementations that communicates with the S-PEPS as well as implementation that handles responses from Middleware Clients (e.g. BKU or BürgerClient)
VIDP-API	This module holds all interfaces and exceptions using the Austrian extension to OpenSAML for all uses of STORKAuthnRequest and STORKResponse. It acts as common module for VIDP-EJB, VIDP-WS-EJB, and VIDP-V-SPEPS-WEB

VIDP-SPWare-Client-Germany-EJB	Contains a local stateless, none transaction session implementation that encapsulates the German SPWare client. It could be plug-in at anytime hence use of service locator to discover it.
VIDP-SPWare-Client-Austria-EJB	Contains a local stateless, none transaction session implementation encapsulates the Austria SPWare client. It could be plugged-in at anytime hence use of service locator to discover it.
VIDP-PEPSConnector-WEB	Handles responses from C-PEPS.
VIDP-SPMWAdapter-WEB	This module provides a web interface for Austrian legacy applications to support STORK authentication functionality.

Table 28: Description of VIDP Modules

Name	Description	Comments
VIDP-SPWare-Germany-EJB	Contain local, stateless implementation of the German SPWare that handles the mapping and dispatching of request to eID-Service. It also contains an enterprise web service implementation of the SPWare that interacts with the VIDP with it corresponding handlers	
VIDP-SPWare-Austria-EJB	Contain local, stateless implementation of the Austria SPWare that handles the mapping and dispatching of requests to MOA-ID.	
VIDP-SPWare-AT-Web	The web interface the user is redirected to after having been successfully authenticated by MOA-ID	
VIDP-PEPSConnector-EJB	Processes the response from C-PEPS as well as determines the correct C-PEPS for sending the STORKAuthnRequest to.	
Comments		

Table 29: Description SPWare Application Modules

Name	Description	Comments
VIDP-Service-EJB	Contains remote session management implementation, option model, as well as persistence service	
Comments		

Table 30: Description of Service Application Modules

Name	Description	Comments
MW-Util-API	Contains Utility classes such timestamp generators, Attributes Mapper, and Interceptors	
MW-Exception-API	Contains platform standard error handling concept. All other modules that offer external exceptions must extend the MWBusinessException	
MW-Messages-API	Contains messages required by plug-ins/ons such as STORKAuthRequest	
STORKSAMLEngine	Contains all SAML related elements and utilities	
VIDP-SPWare-API	Contains interfaces, exception and messages as well as utilities	
LogService	Use for logging in/out bound messages	

Table 31: Description of Common Modules

4.3 Packages

4.3.1 VIDP-API Module

Package Name	Description
eu.stork.vidp.api.interfaces	Holds interfaces like SPWare, VIDP (local), VIPD-WS (Remote interface), persistence services, PEPSConnectors and Service Locator, etc
eu.stork.vidp.api.exception	All exceptions returned by any MW component
eu.stork.vidp.api.message	All in/out messages consumed or sent by any MW component
eu.stork.vidp.api.messages.saml	Extension to SAMLRequest/Response
eu.stork.vidp.api.util	Utility and helper classes accessed by most VIDP components

Table 32: Packages in VIDP-API Module

4.3.2 VIDP-EJB

Package Name	Description
eu.stork.vidp.impl.vidp	Holds VIDP implementation as stateless session bean
eu.stork.vidp.impl.spwareclient	Holds Service Locator implementation

Table 33: Packages VIDP-EJB Module

4.3.3 VIDP-WS-EJB

Package Name	Description
eu.stork.vidp.impl.ws	Holds VIDP-WS implementation as stateless enterprise web service

Table 34: Packages in VIDP-WS-EJB Module

4.3.4 VIDP-V-SPEPS-WEB

Package Name	Description
eu.stork.vidp.impl.web.vpeps	Holds servlets and any other required classes that receive and process requests from S-PEPS

Table 35: Packages in VIDP-V-SPEPS-WEB Module

4.3.5 VIDP-PEPSConnector-WEB

Package Name	Description
eu.stork.vidp.impl.web.pepsconnector	Holds servlets and any other required classes that receive and process responses from C-PEPS

Table 36: Packages in VIDP-PEPSConnector Web

4.3.6 VIDP-SPWare-Client<CountryName>-EJB

Package Name	Description
eu.stork.vidp.spwareclient.impl.	Holds stateless, session implementation of a branded SPWare interface. It doesn’t implement directly the SPWare interface but its local interface must extend the SPWare interface
eu.stork.vidp.vidp.spwareclient.impl .client	Holds clients generated stubs of SPWare web service implementation

Table 37: Packages in VIDP-SPWare-Client<CountryName>-EJB Module

4.3.7 VIDP-Services

Package Name	Description
eu.stork.vidp. impl.persistence.entities	Holds all persistence entities within the VIDP
eu.stork.vidp.impl.persistence.service	Holds implementations that provides services to persistence
eu.stork.vidp.impl.services	Holds remote service implementations like session service, option model etc

Table 38: Packages in VIDP-Services Module

4.3.8 VIDP-SPWare-API

Package Name	Description
eu.stork.vidp.spware.api.interfaces	Holds interfaces like SPWare,
eu.stork.vidp.spware.api.exception	All exceptions returned by SPWare
eu.stork.vidp.spware.api.message	All in/out messages consumed or sent by any SPWare component
eu.stork.vidp.spware.api.messages.saml	Extension to SAMLRequest/Response
eu.stork.vidp.spware.api.util	Utility and helper classes accessed by most SPWare components

Table 39: Packages in VIDP-SPWare-API Module

4.3.9 VIDP-SPWare-<CountryName>-EJB

Package Name	Description
eu.stork.vidp.spware.impl	Holds local stateless implementation of SPWare. Direct implementation of SPWare interface is not allowed. Every MS should have a branded SPWare interface. Therefore an extension of the SPWare interface is made.
eu.stork.vidp.spware.impl.ws	Holds enterprise web service implementation of the SPWare that receives and forwards calls to the local stateless implementation

Table 40: Packages in VIDP-SPWare-<CountryName>-EJB

4.3.10 VIDP-PEPSConnector-EJB

Package Name	Description
eu.stork.vidp.pepsconnector.impl	Implements the back-end functionality of the PEPSConnector

Table 41: Packages in VIDP-PEPSConnector-EJB

4.3.11 SAML Engine

Package Name	Description
eu.stork.vidp.messages.saml	Extensions for STORK to SAML elements
eu.stork.vidp.messages.saml.impl	Implementation of SAML extensions
eu.stork.vidp.messages.stork	STORK specific message elements
eu.stork.vidp.messages.stork.impl	Implementation of STORK specific elements
eu.stork.vidp.messages.util	Utitly classes for e.g. constructing STORK SAML messages

Table 42: Packages in SAML Engine

5 Security Concept

The general security principles documented in D5.8.3d [6] are the foundation of all designs and developments performed in STORK, thus also for the V-IDP. This chapter discusses security issues concerning the V-IDP and the implemented approach.

5.1 Authentication and Authorization at VIDP

The following will need to authenticate themselves at STORK MW (VIDP)

- SP
- S-PEPS
- C-PEPS

Two authentication approaches can be followed:

- Do authentication before sending functional message.

  This could require simple password authentication or mutual authentication (using PKI). This approach is not used in STORK, thus there is no need for details here.

- Send authentication credentials within the functional message.

  This might require password authentication or usage of other SP unique properties transmitted within the message. At the moment STORK supports SP unique properties like SPID.

5.2 Message Security

In- and outbound messages will need to be signed and encrypted by all systems integrated by VIDP. Since the VIDP acts as a gateway providing services to many SP and S-PEPS, this leads to four major nodes that will require message security. These nodes include:

1. S-PEPS-VIDP (encryption: currently not supported on the message level[2]. Encryption needs to be performed on the transmission layer, e.g. SSL)
2. SP-VIDP (encryption: supported, optional)
3. VIDP-C-PEPS (encryption: currently not supported on the message level. Encryption needs to be performed on the transmission layer, e.g. SSL)
4. VIDP-SPWare (Webservice-Interface) (encryption: supported, optional)

For item 1 and 3 it will be followed the defined STORK message security approach. For item 2 and 4 there is no real message security approach.

Deviations

- Germany

  In the case of the eID-Service whose message security approach has no consideration of a broker/gateway approach, special care must be undertaken to directly integrate the current state of the eID-Service to STORK. SP message security credentials are mapped directly within the eID-Service. A message from SP to eID-Service will pass two nodes with

[2] The V-IDP supports SAML encryption, but the STORK interface specification does not make use of that feature due to existing legal requirements in some countries.

message security (SP-VIDP and VIDP-EID-Service). For the communication between SP and VIDP, the VIDP operator needs to agree on the message security credentials with the SP. In addition, the SP message security credentials towards the eID-Service must be handled by the V-IDP instead of the SP. The V-IDP operator thus needs to have a trust relationship with the SP as well as with the eID-Service. The SP needs to entrust the message security handling to the V-IDP and the eID-Service must be made aware that he will have the V-IDP as its communication partner instead of the SP. This includes at least the so called token certificates. Details about the needed configuration for an SP to perform authentication with a German citizen can be obtained from the installation and configuration manual of the V-IDP [12].

- Austria

  For communication with the national Austrian middleware modules message security is achieved via transport level security (SSL/TLS).

6 Codes and Attributes (VIDP-WS and VP)

This section presents the various codes used in controlling the authentication process.

6.1 Status (VIDP-WS)

Code	Message	Description
0	Success	Authentication was successful
1	Failed	Authentication failed
2	Pending	Authentication is still in processing (Not supported at the moment)

Table 43: Status (VIDP-WS)

6.2 Error Codes

Previously, the STORK error codes were quite few. We follow the approach of providing as detailed and quite distinct error codes to clients. This will require allots of error codes which can be realised through categorization as well as modular levels. In an upcoming version, the error codes shall be harmonized with the newly updated STORK error codes list which is now at a comparable level of detail.

The following list includes the internal V-IDP error numbers, which are mapped to the generic STORK error numbers.

Error Code	Message	Description	Client's reaction
	Session and SP related error codes		
100	Invalid SessionID	Invalid SessionID	3
101	SessionID not Found	No SessionID found	6
102	Session Ended	Session in End state	6
103	Missing SessionID Parameter	Missing SessionID Parameter in request	6
104	Missing SessionID Parameter Value	End User SessionID Parameter has value	3
105	Invalid SPID	Invalid SessionID	3
107	Missing SPID Parameter	Missing SPID Parameter in request	6
108	Missing SPID Parameter Value	End User SPID Parameter has value	3
109	Invalid TransactionID	Invalid TransactionID	3
110	TransactionID not Found	TransactionID not found	6
111	Transaction Ended	Transaction in finished state	6
112	Invalid Country	Invalid Country	3
113	Country not Found	No Country found	6
114	Missing Country Parameter	Missing CountryParameter in request	6
115	Missing Country Parameter Value	End User Country Parameter has value	3
117	Invalid Assertion Consumer URL not Found	No Assertion Consumer URL found	6
118	Missing Assertion Consumer URL Parameter	Missing Assertion Consumer URL Parameter in request	6
119	Missing Assertion Consumer URL Parameter Value	End User Assertion Consumer URL Parameter has request	3
120	Invalid Domain Name	Invalid Domain Name	3
121	Missing Domain Name Parameter	Missing Domain Name Parameter in request	6
122	Missing Domain Parameter Value	Missing Domain Name Parameter has value	3
123	Missing Application Name Parameter	Missing Application Name Parameter in request	6
124	Missing Application Name Parameter Value	End User Application Name Parameter has value	3
125	Invalid Application Name	Invalid Application Name	
126	Invalid STORK Attribute Name Parameter	Invalid STORK Attribute Name	6
127	Missing STORK Attribute Parameter Value	Missing STORK Attribute Name Parameter has value	3
128	Missing STORK Attribute parameter	Missing STORK Attribute Name Parameter in request	6
129	Missing STORK Assertion Parameter Value	Missing STORK Assertion Parameter has value	3
130	Missing STORK Assertion parameter	Missing STORK Assertion Parameter in response	6
131	Missing issuerURL Parameter	Missing issuerURL Parameter in request	6
132	Missing issuerURL Parameter Value	End User issuerURL Parameter has value	3
133	Invalid issuerURL	No issuerURL found	6
134	Missing timestamp Parameter	Missing timestamp Parameter in request	6
135	Missing timestamp Parameter Value	End User timestamp Parameter has value	3
136	Invalid timestamp	Invalid timestamp	6
	S-PEPS		
152	S-PEPS not Found	S-PEPS not found	
153	S-PEPS not enabled	S-PEPS not in enabled state	
	C-PEPS		
161	Invalid C-PEPSID	No matching with adopted pattern	
162	C-PEPS not Found	C-PEPS not found	
163	Missing C-PEPSID Parameter	Request has not C-PEPS parameter	
164	Missing C-PEPSID Value	C-PEPS parameter has not value	
165	Failed connection to C-PEPS	Failed connection to C-PEPS	
166	C-PEPS General Application Error	C-PEPS General Application Error	
167	Failed Authentication to C-PEPS	Failed Authentication to C-PEPS	
	SPWare		
168	Unkonw SPWare	Unkonw SPWare	
169	Not Enabled	SPWare Not Enabled	
170	Can't connect to SPWare	Can't connect to SPWare	
172	SPWare Configuration Error	SPWare Configuration Error	
	SPWare-Memberstate IDP		
176	Failed connection to MS-IDP	Failed connection to MS-IDP (eg. IDService)	
177	Application error from MS-IDP	Application error from MS-IDP	
178	SPWare-MS IDP Connection timeoutand MS-IDP	SPWare-MS IDP Connection timeoutand MS-IDP	
179	MS IDP General Application Error	MS IDP General Application Error	
180	Invalid MS IDP URL	Configured URL is Invalid	
	MS IDP-Client (BC)		

Error Code	Message	Description	Client's reaction
185	MS-IDP-BC Connnection timout between IDP and BC	MS-IDP-BC Connnection timout between IDP	
186	BC can't connect to MS IDP	BC can't connect to MS IDP	
187	BC is not started	BC is not started	
188	User cancelled authentication	User cancelled authentication	
189	User denied attribute retrieval	User denied attribute retrieval	
	SP Account related error codes		
200	Invalid SP Account	SP pattern is used else default pattern	3
202	Invalid Password	Password invalid	3
203	Account not Found	Account not found	6
204	Blocked Account	Account in blocked state	6
205	Disabled Account	Account in disabled state	6
206	Deactivated Account	Account in deactivated state	6
207	Suspended Account	Account in suspended state	6
208	Deleted Account	Account in deleted state	6
209	Missing Username Parameter	Missing Username Parameter in request	3
210	Missing Password Parameter	Missing Password Parameter in request	3
211	Missing Username Parameter Value	Username Parameter has no value	3
212	Missing Password Parameter Value	Password Parameter has no value	3
213	Unsupported Account State	Account state not defined in Spec	4
	General Communication General Communication error codes		
300	Invalid request	Request is general invalid based on technical spec	3
301	Missing Parameter	Missing parameter is request	3
302	Invalid Parameter	Included parameter not supported by the API	3
303	Missing Parameter Value	Parameter has not value	3
304	General API Error	General API	1
305	Service not available	Service not reachable	1
306	Invalid Timestamp	Timestamp not supported	4
307	Missing Timestamp Parameter	No timestamp element in request	3
308	Invalid TransactionId	Invalid transactionId	4
309	Missing TransactionId Parameter	Missing TransactionId Parameter	3
310	Invalid Policy Version	Invalid policy version. Not supported by SPAuthenticationService	1
311	Missing Policy Version Parameter	No policy version element in request	3
312	Missing Version Parameter Value	No policy version value in version element missing	3
	Invalid PolicyURL	Invalid PolicyURL	3
	Missing PolicyURL Parameter	Missing Policy URL Parameter	3
	Missing PolicyURL Parameter Value	Missing Policy URL Parameter Value	3
	Authentication and Authorization by SP at	Middleware	
400	SP Failed Authentication	General authentication failed	1
401	SP Failed Authorization	General authorization failed	1
402	SP not Found	SP not found	1
403	SP Restriction to Service	SP not allowed to use the service called	2
404	SP Restriction to Service Method	SP not allowed to use the service method called	2
405	Missing SP URL for Notifications	No SP notification URL configured	1
407	Access Denied	Access denied	1
	PEPSConnector MG and its Component		
600	CCC not specified	No citizen country code specified	3
601	PEPS country not supported	PEPS country not supported	6
602	No PEPS destination found	No PEPS destination found	1
603	No AssertionConsumerService URL found	ACS URL of PEPSConnector not defined	1
604	No issuer name for PEPSConnector configured	PEPSConnector issuer name not configured	1
605	Error signing request	Credentials for signing AuthnRequest not found or signing error	1
606	Error building StartAuthResponse	StartAuthResponse cannot be built	1
607	C-PEPS unknown	C-PEPS unknown	3
608	C-PEPS response not valid	C-PEPS response not valid	6

Error Code	Message	Description	Client's reaction
609	C-PEPS assertion not valid	C-PEPS assertion not valid	6
610	No response stored for sessionID	No response stored for sessionID	0
	SPWare AT MG and its Component		
700	Error providing BKU selection page	Error providing BKU selection page	1
701	Error retrieving authentication data from MOA-ID	Error retrieving authentication data from MOA-ID	1
702	Error building STORK response	Error building STORK response	1
	VIDP and Resources MG and its Component		
900	Transaction Timeout	Transaction timeout	1
901	Internal Error	Internal VIDP error	1
902	VIDP Error	VIDP application error	1

Table 44: Error Codes

Clients Reaction Code	Description
0	Try Again
1	Call VIDP Support 0049 ******32
2	Contact VIDP Customer Service 0049 ******31
3	Try Again with right parameters and value
4	See VIDP-WS API Specification Document
5	No reaction
6	No Retry

Table 45: Clients Reaction Code

6.3 Current Supported Attributes by eID-Service

These attributes are not to be maintained within this specification. To have a full understanding of it the BSI TR-03110 of the Personalausweis should be consulted.

Attribute Name	Data group or Function	Content of Attribute value	Type of Attribute Value in Authentication response	Type of Attribute Value in Authentication request
DocumentType	DG1	Dokumententyp, „ID“ für nPA	string	-
IssuingState	DG2	Ausgebender Staat, „D“ für Deutschland	string	-
DateOfExpiry	DG3	Ablaufdatum	string	-
GivenNames	DG4	Vornamen	string	-
FamilyNames	DG5	Familienname(n)	string	-
ArtisticName	DG6	Ordensname/ Künstlername	string	-
AcademicTitle	DG7	Doktorgrad	string	-
DateOfBirth	DG8	Geburtsdatum	string	-

Attribute Name	Data group or Function	Content of Attribute value	Type of Attribute Value in Authentication response	Type of Attribute Value in Authentication request
PlaceOfBirth	DG9	Geburtsort	string	-
PlaceOfResidence	DG17	Adresse	string	-
RestrictedIdentification	Restricted Identification	Sektorspezifische Kennung (Pseudonym)	string	-
CommunityIdVerfication	Community ID Verification	Ergebnis bzw. Anfragewert der Vergleichsfunktion Wohnortabfrage (amtlicher Gemeindeschlüssel)	string	xs:string .
AgeVerification	Age Verification	Ergebnis bzw. Anfragewert der Vergleichsfunktion Altersverifikation	string	xs:string .
DocumentValidity	Gültigkeits-prüfung 2)	Ergebnis der Gültigkeitsprüfung des Dokumentes	string	-
Comments	The following attributes (CommunityIdVerification, AgeVerification, DocumentValidity) for simplicity should be used as QueryAttribute if information concerning them has to be transmitted. Otherwise other attributes uses NormalAttributes during initAuthentication			

Table 46: Current Supported Attributes by eID-Service

Attribute Condition

Attribute Condition	Description	Comments
Mandatory	Indicates that provision of requested attribute is mandatory to end-user	Very important for SP flow
Optional	Indicates that provision of requested attribute is optional to end-user	

Table 47: Attribute Condition

6.4 DE eID-Service STORK Attributes[3] Mapping

Some clarification is needed at STORK level before filling this table. Moreover it will require a working session between persons with good mastering in eID-Service attributes as well those with STORK attributes. Any wrong mapping will lead to a malfunctioning of the system. The table below presents a proposed mapping based on the document provided by STORK. However, a future review by all parties is required an additional table: eID-Service STORK Attributes Mapping.

eID-Service Attribute Name	STORK Attribute
DocumentType	N/A
IssuingState	N/A
DateOfExpiry	N/A
GivenNames	givenName
FamilyNames	surname

[3] For a complete list and description of attributes, see [1]

ArtisticName	pseudonym
AcademicTitle	title
DateOfBirth	dateOfBirth
PlaceOfBirth	N/A
PlaceOfResidence	textResidenceAddress
RestrictedIdentification	eIdentifier
CommunityIdVerfication	N/A
AgeVerification	isAgeOver
DocumentValidity	N/A
General mapping comments	*As can be seen above, there are a list of attributes which cannot be mapped to STORK attributes. Currently, these are not forwarded when sent over the V-PEPS plug-on.*

Table 48: Service STORK Attributes Mapping

6.5 AT STORK Attributes Mapping

This section describes the mapping of the attributes delivered by the Austrian middleware MOA-ID to the attributes defined by STORK. MOA-ID only issues a certain amount of user attributes thus not all STORK attributes can be supported. Table 49 illustrates this mapping.

MOA-ID Attribute Name	STORK Attribute
NameIdentifier	http://www.stork.gov.eu/1.0/eIdentifier
GivenName	http://www.stork.gov.eu/1.0/givenName
FamilyName	http://www.stork.gov.eu/1.0/surname
DateOfBirth	http://www.stork.gov.eu/1.0/dateOfBirth

Table 49: MOA-ID STORK Attributes Mapping

6.6 Attribute Status

These are status codes specifically for any attribute requested by the SP. Currently, the eID-Service does not support such status attributes at individual attribute scope rather global at request level.

Code	Message	Description
0	Success	Attribute successfully retrieved
1	Failed	No reason provided
2	SP not authorised	SP does have read access to the attribute. Not yet supported, 1 will be returned
3	End user denied	End user refused retrieval of attribute. Not yet supported, 1 will be returned
4	Attribute not supported	End user does not have the requested attribute. Not yet supported, 1 will be returned

Table 50: Attribute Status

7 Persistence

For the Middleware to function properly there is a need to configure the system based on the contract with respective systems such as S-PEPS, C-PEPS and SP. Other configurations such as options are used to manage runtime issues such as switching the system to a staging or live system as well as managing sessions. The following sections provide a detailed description of these tables.

7.1 State Table

ID	Value	Description
1	Enabled	Entry is active. Only domain objects with this state can be read on runtime.
2	Disabled	Entry is not active. Entry can be changed to Enabled state again.
3	Deleted	State can never be changed to active state again, so it can later be deleted completely.
4	Inactive	Any newly configured object that requires validation or approval or longtime configuration should by default have this state.

Table 51: State Table

7.2 SP

Holds Information for SPs. Default values may be configured and will thus lead to an automatic inclusion of this parameter if omitted in the request (optimization reasoning).

Name	Description	Type	Default
SPID	Identifies an SP that can use VIDP	Varchar, not null Unique, primary key	Values from Sequence named SEQ_SP
DomainName	Indicates domain of an SP	Int, not null, unique	
ApplicationName	Indicates the human readable application name of an SP	Varchar, not null	
StateID	Indicates state of the SP. Reference to state table.	Int, not null	
CountryID	Indicates country of SP. References the Country table	Int, not null	
AssertionConsumerServiceUrl	Indicates the URL where UserAgent (eg. AusweisApp) can redirect to SP after connecting to SPWare (eID-Service)	Varchar, not null	Optional, may serve as additional security check if used to compare to attribute in AuthnRequest
PubSignCert	SP's certificate used by VIDP to validate all SPs' signed requests	Varchar, null	Null, can be sent during request
Pub EnryptCert	SP's certificate used by VIDP to encrypt all SPs' responses	Varchar, null	Null, can be sent during request
PrSignCert	SP's certificate used by VIDP to sign all SPs' responses	Varchar, not null	
PrEncryptCert	SP's certificate used by VIDP to decrypt all SPs' requests	Varchar, not null	

Name	Description	Type	Default
MaxRequestPerMinute	Number of request per minute to handled DDOS (conversations or Requests?)	Varchar, not null	5
CreatedBy	Holds name of person who created that entry	Varchar, not null	
DateCreated	When this SP was created	timestamp NULL default NULL	Null (since SP are created by persistence layer this value will never be null else is a faked one)
ModifiedBy	Holds name of person who last modified that entry	Varchar, null	Null
LastModified	When this SP was last modified	Timestamp, null	CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP

Table 52: SP Configuration

7.3 S-PEPS

Holds Information for S-PEPS

Name	Description	Type	Default
ID	Identifies an S-PEPS that can use VIDP	Varchar, , Not null Unique, primary key	Values from Sequence named SEQ_SPEPS
Name	Indicates domain of an SP	Int, not null, unique	
AssertionConsumerServiceUrl	Indicates the URL where VIDP can redirect responses from SPWare to S-PEPS	Varchar, not null	
CountryID	Indicates country of S-PEPS. References the Country table	Int, not null	
PubSignCert	S-PEPS certificate used by VIDP to validate its signed requests	Varchar, not null	
Pub EnryptCert	S-PEPS's certificate used by VIDP to encrypting its responses (currently not supported)	Varchar, not null	
PrSignCert	SPEPS's certificate used by VIDP to sign all S-PEPSs' responses	Varchar, not null	
PrEncryptCert	S-PEPS's certificate used by VIDP to decrypt all S-PEPSs' requests (currently not supported)	Varchar, not null	
StateID	Indicates state of this SP. References the State table	Int, not null	
CreatedBy	Holds name of person who created that entry	Varchar, not null	
DateCreated	When this SP was created	timestamp NULL default NULL	Null (since SPEPS are created by persistence layer this value will never be null else is a faked one)

Name	Description	Type	Default
ModifiedBy	Holds name of person who last modified that entry	Varchar, null	null
LastModified	When this SPEPS was last modified	Timestamp, null	CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP
ServiceProviderIssuerURL	The issuer name of the S-PEPS	Varchar, not null	

Table 53: PEPS Configuration

7.4 C-PEPS

VIDP uses this table when communicating with C-PEPS.

Name	Description	Type	Default
ID	Identifies an C-PEPS that can use VIDP (issuer element)	Varchar, , Not null Unique, primary key	Values from Sequence named SEQ_CPEPS
Name	Indicates domain of an C-PEPS	Int, not null, unique	
CPEPSURL	Indicates the URL of the C-PEPS where VIDP can forward request	Varchar, not null	
CountryID	Indicates country of C-PEPS. References the Country table	Int, not null	
PEPSConnector	Hold className for the PEPSConnector implementation.	Int, not null	
PubSignCert	C-PEPS certificate used by VIDP to validate its signed responses	Varchar, not null	
Pub EnryptCert	C-PEPS's certificate used by VIDP to encrypting its requests (currently not supported)	Varchar, not null	
PrSignCert	C-PEPS's certificate used by VIDP to sign all the C-PEPSs' responses	Varchar, not null	
PrEncryptCert	C-PEPS's certificate used by VIDP to decrypt all the C-PEPSs' requests (currently not supported)	Varchar, not null	
Version	Indicates version of the C-PEPS implementation	Int, not null	1
StateID	Indicates state of this C-PEPS. References the State table	Int, not null	
CreatedBy	Holds name of person who created that entry	Varchar, not null	

Name	Description	Type	Default
DateCreated	When this SP was created	timestamp NULL default NULL	Null (since CPEPS are created by persistence layer this value will never be null else is a faked one)
ModifiedBy	Holds name of person who last modified that entry	Varchar, null	null
LastModified	When this CPEPS was last modified	Timestamp, null	CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP
ServiceProviderIssuerURL	The issuer name of the C-PEPS	Varchar, not null	

Table 54: C-PEPS Configuration

7.5 SPWare

Holds information for the SPWare

Name	Description	Type	Default
ID	Identifies of a SPWare	Varchar, , Not null Unique, primary key	Values from Sequence named SEQ_SPWare
CountryID	Indicates country of SPWare. References the Country table	Int, not null	
StateID	Indicates state of this SPWare. References the State table	Int, not null	
CreatedBy	Holds name of person who created that entry	Varchar, not null	
DateCreated	When this SPWare was created	timestamp NULL default NULL	Null (since SPWare are created by persistence layer this value will never be null else is a faked one)
ModifiedBy	Holds name of person who last modified that entry	Varchar, null	null
LastModified	When this SPWare was last modified	Timestamp, null	CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP

Table 55: SPWare Configuration

7.6 PEPSConnector

Holds information for the PEPSConnector.

Name	Description	Type	Default
ID	Identifies the PEPSConnector (issuer)	Varchar, , Not null Unique, primary key	
AssertionConsumerServiceURL	URL where PEPSConnector wants to receive C-PEPS responses	Varchar, not null	

Name	Description	Type	Default
StateID	Indicates state of the PEPSConnector. References the State table	Int, not null	
CreatedBy	Holds name of person who created that entry	Varchar, not null	
DateCreated	When this PEPSConnector was created	timestamp NULL default NULL	
ModifiedBy	Holds name of person who last modified that entry	Varchar, null	null
LastModified	When this PEPSConnector was last modified	Timestamp, null	CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP

Table 56: PEPSConnector Configuration

7.7 STORKMemberStates

Holds information for all Member States integrated with the VIDP

Name	Description	Type	Default
CountryID	Identifies a country	Varchar, , Not null Unique, primary key	Values from Sequence named SEQ_Country
Name	Name of country	Int, not null	
StateID	Indicates state of this country entry. References the State table	Int, not null	
CreatedBy	Holds name of person who created that entry	Varchar, not null	
DateCreated	When this country entry was created	timestamp NULL default NULL	Null (since Country entry is created by persistence layer this value will never be null else is wrongly created)
ModifiedBy	Holds name of person who last modified that entry	Varchar, null	null
LastModified	When this country was last modified	Timestamp, null	CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP

Table 57: STORKMemberStates Configuration

7.8 Session

Holds session information during a complete transaction (InitAuthentication and getAuthenticationData). Contains information about all active sessions. In final state, the session is deleted and copied to session history.

Name	Description	Type	Default
SessionID	Identifies the session	Varchar, Not null Unique, primary key	Values from Sequence named SEQ_SESSION

Name	Description	Type	Default
SPID	SP identification. References SP table	Int, Not null	
TransactionID	TransactionID used by VIDPWS and SP (ID element of SAML message)	Varchar, null	Null
SPWareURL	URL used during this session to connect to a SPWare e.g. eID-Service	Varchar , Not null	
SPWareRefreshURL	Used to connect to SPWare during getAuthenticationData	Varchar, null	Null
S-PEPSAssertionConsumerURL	Used during this session to redirect response to S-PEPS	Varchar, null	Null
C-PEPSURL	Used during this session to connect to C-PEPS	Varchar, null	Null
SPAssertionConsumerURL	Used during this session where end user can send request to SP which will then call VIDP-WS to get authentication data	Varchar, null	Null
AuthProcessingStatusID	Indicates the status of authentication process. 0 =SUCESS, 1=FAILURE, 2=PENDING	Int, , Not null	
Caller	Indicates who the caller is P=S-PEPS S=SP	Char , Not null	P
ForwardedTo	Indicates where the call is been forwarded: C=C-PEPS W=SPWare	Char, Not null	W
RequestedAttributes	The RequestedAttributes subtree as appearing in the StorkAuthnRequest	Varchar, Not null	
ResponseAttributes	The Attributes element of the Response message without the values	Varchar, null	
MaxNumberOfRetries	Max number of retries for a conversation for this specific SP or SPEPS	Int, Not null	5
RetriesCount	Number of retries for this specific session	Int, Not null	0
PollingEnabled	If polling to get authentication data at SPWare (eID-Service) or PEPSConnector is enabled	Boolean null	
MaxPollingNumber	Max number of polls for a session at a SPWare (e.g. eID-Service) or C-PEPS	Int , null	
PollRetriesCount	Number of times poll at SPWare or C-PEPS within this session	Int , null	
FailureID	STORK errorcode, holds value only when an error occured within VIDP or is returned from SPWare. Or C-PEPS It references STORKErrorCode table	Int, null	Null

Name	Description	Type	Default
FailureMessage	STORK error message. Holds value only when an error occurred within VIDP or is returned from SPWare or C-PEPS. It references STORKErrorCode table	Varchar, null	Null
SPWareErrorCodeID	Reference SPWareErrorCode table, and holds value only if SPWare returned an error	Int , null	Null
SPWareFailureComment	Message returned by SPWare giving reasons for failure. Holds value only if SPWare returned an error	Varchar, null	Null
CPEPSErrorCodeID	Reference STORKErrorCode table, and holds value only if C-PEPS returned an error	Int , null	Null
CPEPSFailureComment	Message returned by C-PEPS giving reasons for failure. Holds value only if C-PEPS returned an error	Varchar, null	Null
SPCertSig	The certificate used to validate SP request signature if session was between SP.	Varchar, null	
SPCertEnc	The certificate used to encrypt SP response if session was between SP	Varchar, null	
CPEPSCertSig	The certificate used to validate C-PEPS response signature if session was between C-PEPS.	Varchar, null	
CPEPSCertEnc	The certificate used to encrypt C-PEPS request if session was between C-PEPS	Varchar, null	
SPEPSCertSig	The certificate used to validate S-PEPS request signature if session was between S-PEPS.	Varchar, null	
SPEPSCertEnc	The certificate used to encrypt S-PEPS response if session was between S-PEPS	Varchar, null	
SPWareCertSig	The certificate used to validate SPWare response signature if session was between SPWare.	Varchar, null	
SPWareCertEnc	The certificate used to encrypt SPWare request if session was between SPWare	Varchar, null	
StateID	Indicates state of this session. References the State table	Int, not null	
CreatedBy	Holds name of persistence implementation used by VIDP	Varchar, not null	
DateCreated	When this session was created	Timestamp NULL default NULL	Null (since sessions are created by persistence layer this value will never be null else is a faked one)
ModifiedBy	Holds name of persistence implementation used by VIDP	Varchar, null	Null

Name	Description	Type	Default
LastModified	When this session was last modified	Timestamp, null	CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP

Table 58: Session Tracking Table

7.9 Session History

The session history table reflects a copy of the session table and holds session information of finished transactions. For details on the table elements see section 0.

7.10 STORKAttributes

Holds STORK specific attributes.

Name	Description	Type	Default
AttributeID	Identifies an attribute	Varchar, , Not null Unique, primary key	Values from Sequence named SEQ_STORKAttibute
Name	Name of attribute	Varchar, not null	
Description	Description of attribute	Varchar, not null	
Type	Indicate data type of the attribute	Varchar, not null	
StateID	Indicates state of this attribute. References the State table	Int, not null	
CreatedBy	Holds name of person who created that entry	Varchar, not null	
DateCreated	When this attribute was created	timestamp NULL default NULL	Null (since attributes are created by persistence layer this value will never be null else is wrongly created)
ModifiedBy	Holds name of person who last modified that entry	Varchar, null	Null
LastModified	When this SP was last modified	Timestamp, null	CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP

Table 59: STORK Attributes Configuration

7.11 CountryAttribute

Holds attributes of member states with mappings to corresponding STORK attributes

Name	Description	Type	Default
AttributeID	Identifies an attribute	Varchar, , Not null Unique, primary key	Values from Sequence named SEQ_CountryAttibute
Name	Name of attribute	Varchar, not null	
Description	Description of attribute	Varchar, not null	
AttributeTypeID	Indicate data type of the attribute by referencing attributetype table	Varchar, not null	

CountryID	Country to which attribute belongs. References STORKAttribute table	Int, not null	
STORKAttributeID	STORK Attribute. References STORKAttributes table	Int, not null	
StateID	Indicates state of this attribute. References the State table	Int, not null	
CreatedBy	Holds name of person who created that entry	Varchar, not null	
DateCreated	When this country attribute was created	timestamp NULL default NULL	Null (since country attribute are created by persistence layer this value will never be null else is wrongly created)
ModifiedBy	Holds name of person who last modified that entry	Varchar, null	Null
LastModified	When this country attribute was last modified	Timestamp, null	CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP

Table 60: CountryAttributes Configuration

7.12 STORKErrorCode

Holds STORK error codes defined in specification

Name	Description	Type	Default
ID	Identifies a STORK error code	Varchar, , Not null Unique, primary key	Value from STORK error code Specification
Name	Name of error code name	Varchar, not null	
Description	Description of the error code	Varchar, not null	
StateID	Indicates state of this error code. References the State table	Int, not null	
CreatedBy	Holds name of persistence implementation used by VIDP	Varchar, not null	
DateCreated	When this error code entry was created	timestamp NULL default NULL	Null (since error codes are created by persistence layer this value will never be null else is wrongly created)
ModifiedBy	Holds name of person who created that entry	Varchar, null	Null
LastModified	When this error code was last modified	Timestamp, null	CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP

Table 61: STORKErrorCode Configuration

7.13 SPWareErrorCode

Holds SPWare error codes with corresponding mapping to STORK error codes.

Name	Description	Type	Default
ID	Identifies a SPWare error code	Varchar, , Not null Unique, primary key	Value from SPWare error code Specification
ErrorCode	SPWare error code	Varchar, not null	
Message	Message for this SPWare error code	Varchar, not null	
Description	Description of the error code	Varchar, not null	
STORKErroCodeID	Mapping to STORK error code. Its references STORKErrorCode t able		
SPWareID	Refenreces the SPWare table. Holds MW SPWare	Int, not null	
StateID	Indicates state of this attribute. References the State table	Int, not null	
CreatedBy	Holds name of person who created that entry	Varchar, not null	
DateCreated	When this SPWare error code was created	timestamp NULL default NULL	Null (since SPWare error codes are created by persistence layer this value will never be null else is wrongly created)
ModifiedBy	Holds name of person who last modified that entry	Varchar, null	Null
LastModified	When this SPWare error code was last modified	Timestamp, null	CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP

Table 62: SPWareErrorCode Configuration

7.14 Option Model Configuration

This section presents the option configuration model of the Middleware. An option that holds global variables is derived from a specific option group which must be a child element of an option type. A unique identification of an option will be an option item which is quite specific to a context such as SPWare, SP, C-PEPS, or S-PEPS.

7.14.1 Option Type

Name	Description
SPWare	Holds information related to SPWare (e.g. eID-Service)

Table 63: SPWare Option Type

7.14.2 Option Group

Name	Description	Option Type
SPWareConnection	Holds connection information related to a SPWare (e.g. eID-Service)	SPWare
Polling	Holds polling information related to a SPWare (e.g. eID-Service)	SPWare

Table 64: SPWare Option Group

7.14.3 Option

Name	Description	Option Group	Default-String	Default Float	Default Bool-ean	Default Date
StageingURL	URL to staging SPWare	SPWareConnection	http://www.eID-Service.de			
LiveURL	URL to live SPWare	SPWareConnection	http://www.eID-Service.de			
Production	Production to connect to(Mock, Staging, or Live)	SPWareConnection	Mock			
PollingEnabled	By PENDING during getAuthentication Data SPWare can keep polling	Pollings			No	
MaxNumberOf Pollings	Max number of pollings	Pollings			5	
ConnectionTim eout	The timeout in seconds for connection	SPWareConnection	30			
SignCert	For signature validation	SPWareConnection				
EncryptCert	For encryption	SPWareConnection				
MessageSignatu reEnabled	If message should be signed	SPWareConnection				
MessageEncrypt ionEnabled	If message should be encrypted	SPWareConnection				
SSLServerAuth Enabled	If SSL server authentication is enabled	SPWareConnection				
SSLClientAuth Enabled	If SSL client authentication is enabled	SPWareConnection				
SSLTrustStore	Truststore to be used for SSL connection	SPWareConnection				
SSLClientKeySt ore	Keystore to be used for SSL client authentication	SPWareConnection				
BkuURL	URL to client middleware	SPWareConnection				
MessageEncodi ng	Encoding type for message	SPWareConnection				
AuthenticationE nabled	If authentication is enabled	SPWareConnection				

Authentication MutualEnabled	If PKI mutual authentication is required	SPWareConnection				
SPWareMockProcessor	SPWare mock implementation used when production is in mock state	SPWareConnection				

Table 65: SPWare Option

7.14.4 Option Item

Used to override option default values for a specific context.

8 References

[1] D5.7.3 Functional Design for PEPS, MW models and interoperability, STORK eID-Consortium, Final Version

[2] D5.8.3 Technical design, eID-Consortium, Final Version

[3] D5.8.3a Software Architecture Design, eID-Consortium, Final Version

[4] D5.8.3b Interface Specification, eID-Consortium, Final Version

[5] D5.8.3c SoftwareDesign for PEPS architecture, eID-Consortium Final Version

[6] D5.8.3d Security Principles and Best Practices, eID-Consortium, Final Version

[7] Schamberger, Karlinger, Moser: Spezifikation MOA ID, Version 1.4, 02.08.2007

[8] Project MOCCA, http://mocca.egovlabs.gv.at/

[9] The Austrian Citizen Card, http://www.buergerkarte.at/konzept/securitylayer/spezifikation/aktuell/

[10] Security Assertion Markup Language (SAML), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[11] The Austrian E-Government Act

[12] Common V-IDP Deployment Instructions, v1.6

A. Appendix Germany Integration

A.1 Overview

Germany decided the introduction of the electronic identification card (Neuer Personalausweis, nPA), which provides the possibility of an electronic identification and authentication of the document holder. This process requires a Software component ("Bürgerclient") for the end-user and an eID-Service, which is required to access the data stored electronically on the document.

The STORK approach requires the possibility of an electronic authentication and identification by the use of an infrastructure provided by the member states.

This appendix presents information about the German eID-approach using the Bürgerclient and the eID-Service and its integration into the STORK architecture.

Moreover this appendix handles both designs of a User-Centric Authentication (UCA) whereby authentication request and response are directed through the user agent by using S-PEPS as well as other various alternatives such as avoiding request/response through user agent by directly accessing the VIDP-WS. Moreover there are scenarios whereby a SP will not like to expose the Authentication Service to end users and vice versa. After a successful authentication some SPs will prefer authentication status notification with authentication data in one call (by S-PEPS) while others will prefer to pull the authentication data (VIDP-WS). In order to achieve this dynamism and flexibility, optional configuration model architecture is required. The above description could be summarized to two clear Use Case defined in the STORK Architecture which include:

6. UC-AU-P-eIdService

7. UC-AU-M-eIdService

The following section provides some basic information about the German eID-approach. Please refer to the technical guidelines of the Bundesamt für Sicherheit in der Informationstechnik (BSI) to get more detailed knowledge about the involved specifications.

German Identity Card

Germany has introduced the electronic identity card beginning on November, the 1st of 2010. The German identity card (nPA) provides the functionality to read the personal data stored electronically in the document. This functionality is intended to provide a secure and trustworthy way of accessing personal data by external application, e.g. Service Providers, which need to identify the counterpart of the communication.Before the data can be accessed, a set of cryptographic protocols must be executed which allow an external application to access these data. The protocols name is Extended Access Control (EAC) and it is documented in [TR 03110]. The software components Bürgerclient and eID-Server are required for the execution of the EAC protocol in a remote manner. These Software components are both set up on the technical guideline TR 03112 (eCard-API-Framework)

A.2 Online Authentication with AusweisApp[4] and eID-Service

The Software architecture discussed in this document is based on the eID-Service Approach that is used for the German identity card. The general approach is a user centric way of the online authentication which can be using SAML 2.0. The following graphic depicts this:

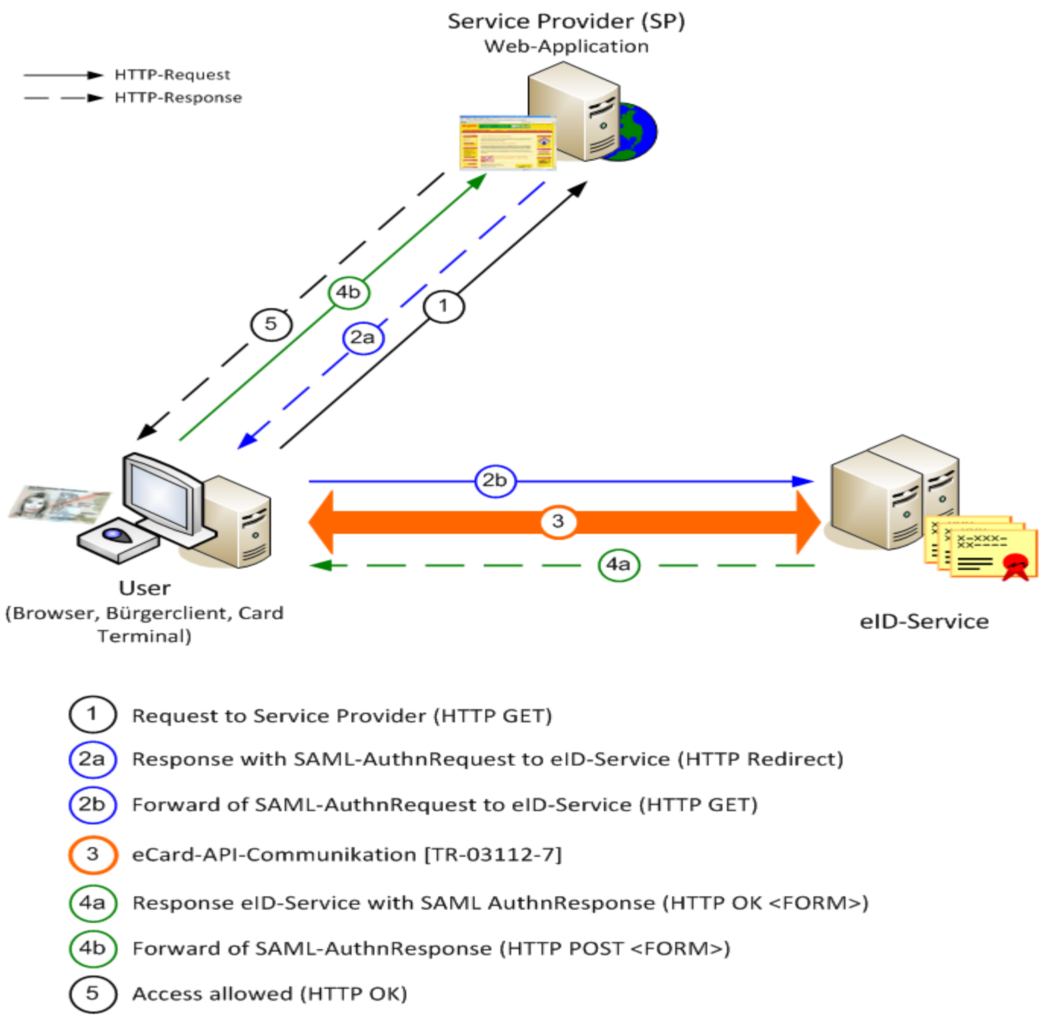

Service Provider (SP)
Web-Application
HTTP-Request
HTTP-Response
1
2a
4b
5
2b
3
4a
User
(Browser, Bürgerclient, Card Terminal)
eID-Service
1 Request to Service Provider (HTTP GET)
2a Response with SAML-AuthnRequest to eID-Service (HTTP Redirect)
2b Forward of SAML-AuthnRequest to eID-Service (HTTP GET)
3 eCard-API-Communikation [TR-03112-7]
4a Response eID-Service with SAML AuthnResponse (HTTP OK <FORM>)
4b Forward of SAML-AuthnResponse (HTTP POST <FORM>)
5 Access allowed (HTTP OK)

Figure 8: German Online Authentication with eID-Service and AusweisApp

In the following the authentication mechanism is explained in a general manner.

The Service Provider (SP) provides a Web-Application containing a resource which requires a user authentication. In case that the user tries to login, the Web-Application generates a SAML authentication request (SAML-AuthnRequest) and sends it back to the Browser together with the directive to forward the SAML-AuthnRequest to the eID-Service.

The eID-Service receives the SAML-AuthnRequest and initiates the communication with the AusweisApp[5]. The AusweisApp and the eID-Service establish a TLS protected channel which is used to read the requested data from the electronic identity card.

[4] Please note that Bürgerclient is an older name for the client middleware now called AusweisApp.

[5] This is already done by sending a response to the browser of the user containing a Mime Object as specified in [eCard-7]. This starts the AusweisApp, which is now connecting to the eID-Service.

When the data has been read, the eID-Service generates the SAML-AuthnResponse and sends it back to the browser which redirects the SAML-AuthnResponse to the Service Provider (SP).

A.3 Use Cases

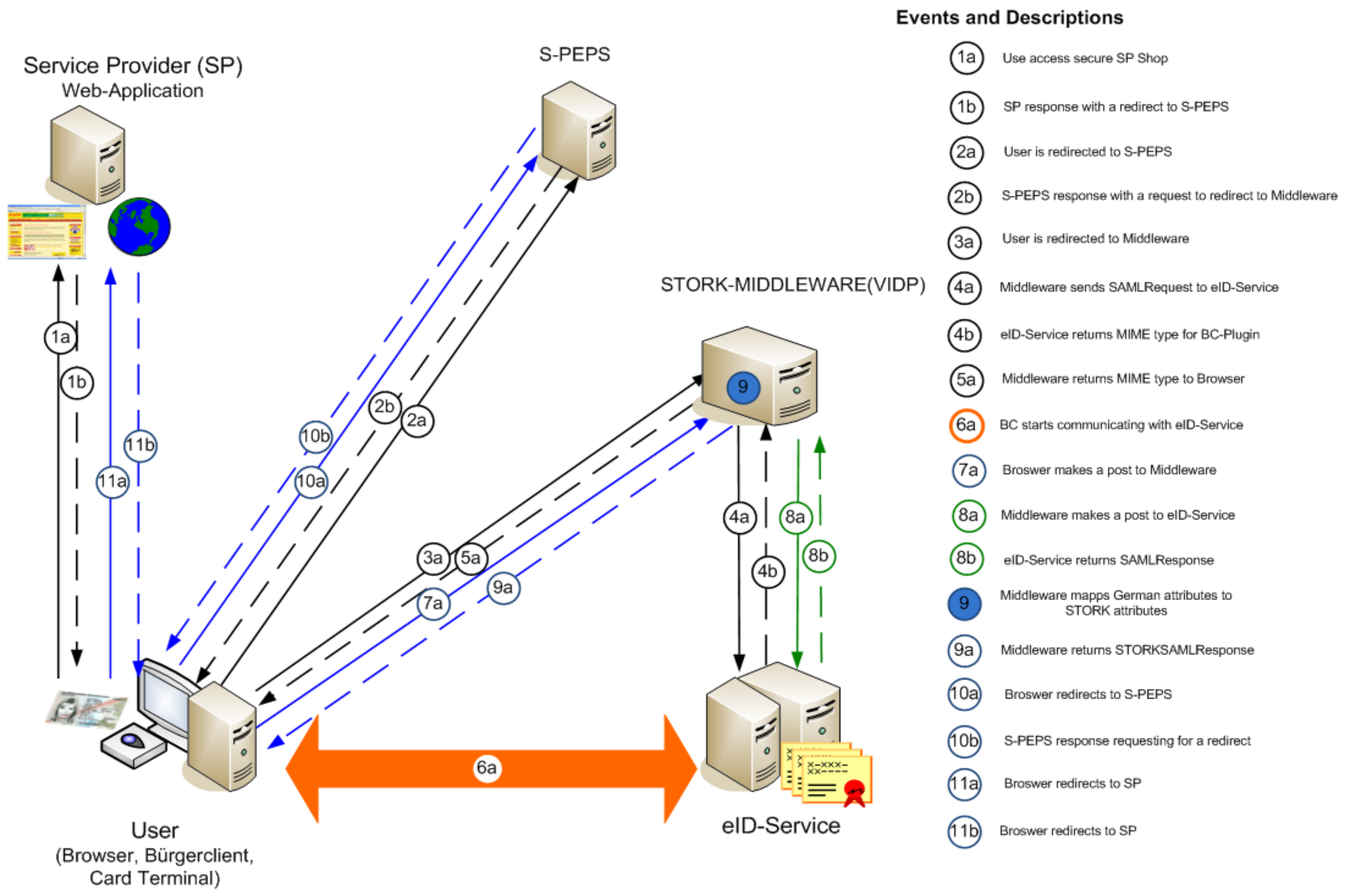
- **UC-AU-P-eIdService**

The following graphic depicts the use case UC-AU-P-eIdService. It shows how the STORK Middleware is triggered by the S-PEPS and how the middleware communicates with the eID-Service used in Germany of Online Authentication with the electronic identity card.

In this use case, all communication is handled user centric. This means that the user agent is a broker for the communication between the participants in the data flow.

The following graphic depicts the data flow between the Service Provider (SP), the S-PEPS, the STORK Middleware (Virtual Identity Provider, VIDP) and the German Online Authentication system consisting of an eID-Service and the AusweisApp.

The role of each component is explained in later chapters.



Events and Descriptions

1a	Use access secure SP Shop
1b	SP response with a redirect to S-PEPS
2a	User is redirected to S-PEPS
2b	S-PEPS response with a request to redirect to Middleware
3a	User is redirected to Middleware
4a	Middleware sends SAMLRequest to eID-Service
4b	eID-Service returns MIME type for BC-Plugin
5a	Middleware returns MIME type to Browser
6a	BC starts communicating with eID-Service
7a	Broswer makes a post to Middleware
8a	Middleware makes a post to eID-Service
8b	eID-Service returns SAMLResponse
9	Middleware mapps German attributes to STORK attributes
9a	Middleware returns STORKSAMLResponse
10a	Broswer redirects to S-PEPS
10b	S-PEPS response requesting for a redirect
11a	Broswer redirects to SP
11b	Broswer redirects to SP

Figure 9: Authentication Flow- UC-AU-P-eIdService

- **UC-AU-M-eIdService**

The following graphic depicts the use case UC-AU-P-eIdService. It shows how the STORK Middleware is triggered by the SP and how the middleware communicates with the eID-Service used in Germany of Online Authentication with the electronic identity card.

In this use case, all communication is handled user agent. This means that the user agent is a broker for the communication between the participants in the data flow.

The following graphic depicts the data flow between the Service Provider (SP), the STORK Middleware (Virtual Identity Provider, VIDP) and the German Online Authentication system consisting of an eID-Service and the AusweisApp. The role of each component is explained in later chapters.

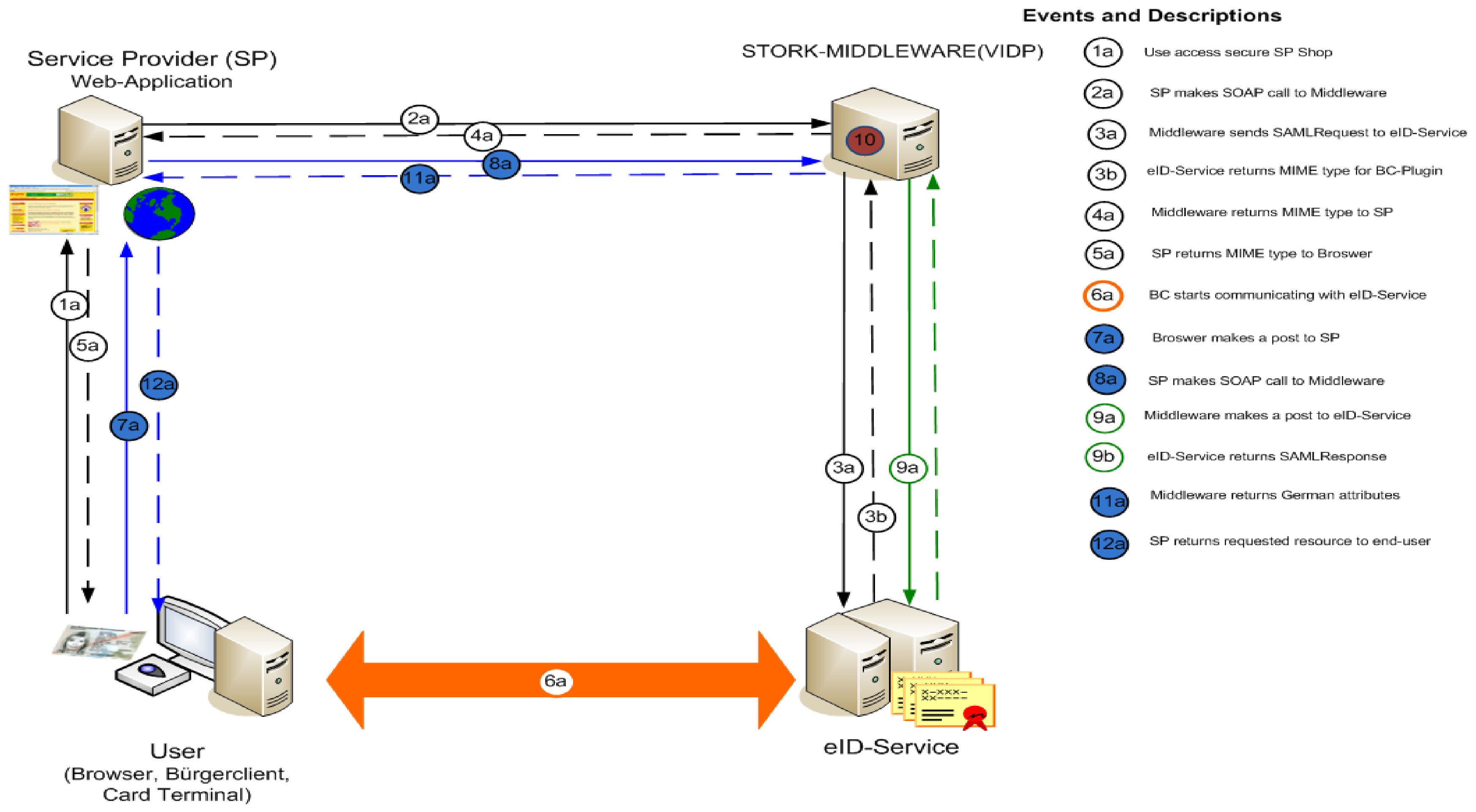

Events and Descriptions
Service Provider (SP)
Web-Application
STORK-MIDDLEWARE(VIDP)
User
(Browser, Bürgerclient,
Card Terminal)
eID-Service
10
1a Use access secure SP Shop
2a SP makes SOAP call to Middleware
3a Middleware sends SAMLRequest to eID-Service
3b eID-Service returns MIME type for BC-Plugin
4a Middleware returns MIME type to SP
5a SP returns MIME type to Broswer
6a BC starts communicating with eID-Service
7a Broswer makes a post to SP
8a SP makes SOAP call to Middleware
9a Middleware makes a post to eID-Service
9b eID-Service returns SAMLResponse
11a Middleware returns German attributes
12a SP returns requested resource to end-user

Figure 10: UC-AU-M-eIdService

A.4 Sequence Diagrams

This section presents all possible use cases supported by the STORK MW.

- **Authentication Flow -UC-AU-P-eIdService**

The S-PEPS handles the complete authentication process with in-outbound messages exposed to the end user. All in/outbound authentication messages are redirected through the end user's browser as illustrated in the figure below followed by a tabled detailing the flow.

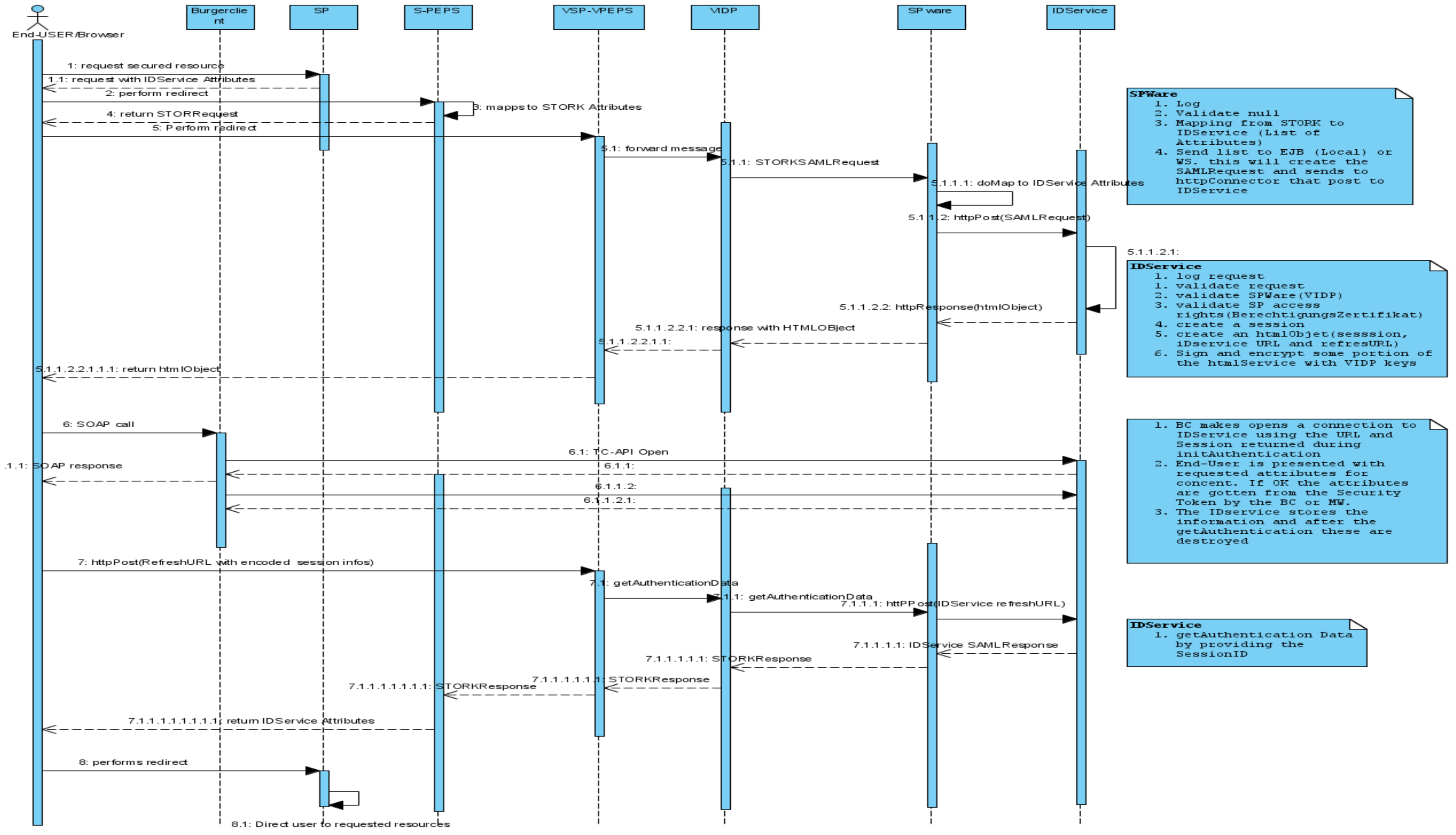

End-USER/Browser
Burgerclient
SP
S-PEPS
VSP-VPEPS
VIDP
SP ware
IDService
1: request secured resource
1.1: request with IDService Attributes
2: perform redirect
3: mapps to STORK Attributes
4: return STORRequest
5: Perform redirect
5.1: forward message
5.1.1: STORKSAMLRequest
5.1.1.1: doMap to IDService Attributes
5.1.1.2: httpPost(SAMLRequest)
5.1.1.2.1:
5.1.1.2.2: httpResponse(htmlObject)
5.1.1.2.2.1: response with HTMLOBject
5.1.1.2.2.1.1:
5.1.1.2.2.1.1.1: return htmlObject
6: SOAP call
6.1: TC-API Open
.1.1: SOAP response
6.1.1:
6.1.1.2:
6.1.1.2.1:
7: httpPost(RefreshURL with encoded session infos)
7.1: getAuthenticationData
7.1.1: getAuthenticationData
7.1.1.1: httPPost(IDService refreshURL)
7.1.1.1.1: IDService SAMLResponse
7.1.1.1.1.1: STORKResponse
7.1.1.1.1.1.1: STORKResponse
7.1.1.1.1.1.1.1: STORKResponse
7.1.1.1.1.1.1.1.1: return IDService Attributes
8: performs redirect
8.1: Direct user to requested resources
SPWare
1. Log
2. Validate null
3. Mapping from STORK to IDService (List of Attributes)
4. Send list to EJB (Local) or WS. this will create the SAMLRequest and sends to httpConnector that post to IDService
IDService
1. log request
1. validate request
2. validate SPWare(VIDP)
3. validate SP access rights(BerechtigungsZertifikat)
4. create a session
5. create an htmlObjet(sesssion, iDservice URL and refresURL)
6. Sign and encrypt some portion of the htmlService with VIDP keys
1. BC makes opens a connection to IDService using the URL and Session returned during initAuthentication
2. End-User is presented with requested attributes for concent. If OK the attributes are gotten from the Security Token by the BC or MW.
3. The IDservice stores the information and after the getAuthentication these are destroyed
IDService
1. getAuthentication Data by providing the SessionID

Figure 11: Sequence Diagram - UC-AU-P-eIdService

Action	Description in the order of events
getResource	End User request for a resource
InitAuthentication	• SP initiates authentication process through S-PEPS by redirecting end-user to S-PEPS with eID-Service Attributes • S-PEPS does logging, authentication and authorization as well as attribute validation • S-PEPS translates request to STORKAuthRequest • S-PEPS does another redirect to VIDP(V-SP/S-PEPS) with STORKRequest • (V-SP/S-PEPS)Calls VIDP with STORKAuthRequest • VIDP decides to route to either a C-PEPS or SPWare. • The VIDP uses the service locator to retrieve the SPWare client that forwards calls to MS SPWare web service implementation • SPWare for the specific country implementation does a mapping to member state attributes and makes use of the SPWare table to access the country MW(e.g. eID-Service access with SAMLRequest signed and encrypted) • An HTTP-Get/SAML request is sent to eID-Service • SPWare Updates session information with response refreshUrl and sessionID from eID-Service • SPWare replaces the refreshUrl in httpobject from the eID-Service with the AssertionConsumerServiceUrl of the S-PEPS • SPWare returns the HttpObject to the VIDP and VIDP to V-SP/V-PEPS) which then returns the call to BC-Plugin
http-POST with refresh URL by BC Plugin	• The BC browser Plug-in calls AusweisApp which opens connection with eID-Service. • The BC browser Plug-in then makes an http-Post using the refreshURL in htmlObject to V-SPEPS/V-SP • The V-SP/S-PEPS receives request from BC-Plug-in logs, and validates message. • V-SP/V-PEPS forwards call to VIDP. • The VIDP determines which SPWare or PEPSConnector to use. • If SPWare chosen and it's Germany SPWare, it uses the sessionID from VIDP to retrieve the refreshURL using the SessionManager. • If authentication processing status is in pending mode, SPWare returns a PENDING Status response to VIDP, checks if polling is enabled globally or for this SP, it will keep polling for the results from eID-Service. Is VIDP allowed to store AuthenticationData after a successful polling? • If authentication is received, the SPWare does mapping to STORKResponse and returns to VIDP. The VIDP returns it to V-SP/V-PEPS which then returns to S-PEPS • S-PEPS maps STORKResponse to eID-Service attributes and return to SP through a redirect

Table 66: General Flow of UC-AU-P-eIdService

- **Authentication Flow UC-AU-M-eIdService**

The VIDP handles the complete authentication process without direct routing of in/outbound messages to end user's browser or BC-plug-in and vice versa. See figure below.

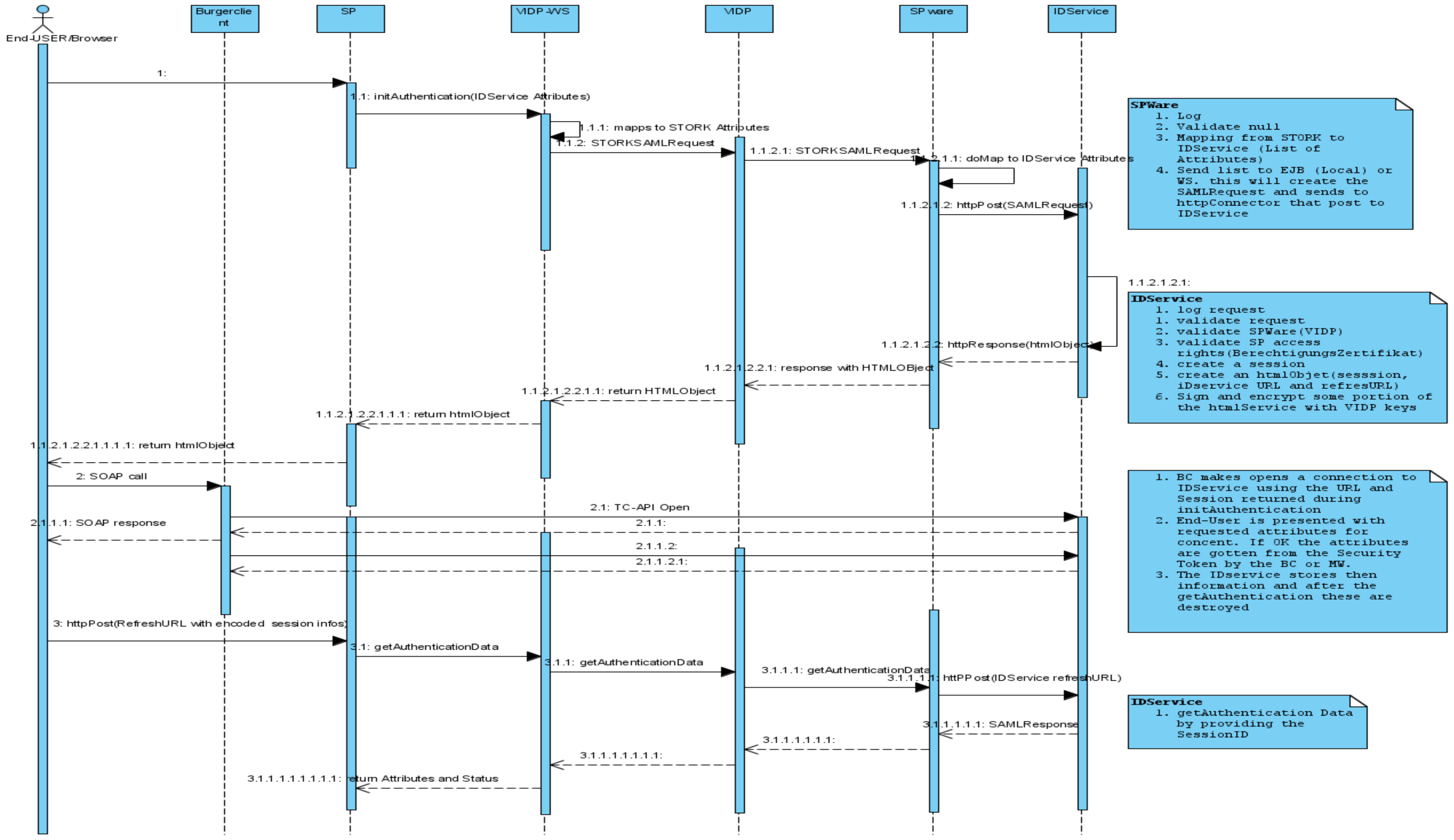

End-USER/Browser
Burgerclient
SP
VIDP-WS
VIDP
SP ware
IDService
1:
1.1: initAuthentication(IDService Attributes)
1.1.1: mapps to STORK Attributes
1.1.2: STORKSAMLRequest
1.1.2.1: STORKSAMLRequest
1.1.2.1.1: doMap to IDService Attributes
1.1.2.1.2: httpPost(SAMLRequest)
1.1.2.1.2.1:
1.1.2.1.2.2: httpResponse(htmlObject)
1.1.2.1.2.2.1: response with HTMLOBject
1.1.2.1.2.2.1.1: return HTMLObject
1.1.2.1.2.2.1.1.1: return htmlObject
1.1.2.1.2.2.1.1.1.1: return htmlObject
2: SOAP call
2.1: TC-API Open
2.1.1:
2.1.1.1: SOAP response
2.1.1.2:
2.1.1.2.1:
3: httpPost(RefreshURL with encoded session infos)
3.1: getAuthenticationData
3.1.1: getAuthenticationData
3.1.1.1: getAuthenticationData
3.1.1.1.1: httPPost(IDService refreshURL)
3.1.1.1.1.1: SAMLResponse
3.1.1.1.1.1.1:
3.1.1.1.1.1.1.1:
3.1.1.1.1.1.1.1.1: return Attributes and Status
SPWare
1. Log
2. Validate null
3. Mapping from STORK to IDService (List of Attributes)
4. Send list to EJB (Local) or WS. this will create the SAMLRequest and sends to httpConnector that post to IDService
IDService
1. log request
1. validate request
2. validate SPWare(VIDP)
3. validate SP access rights(BerechtigungsZertifikat)
4. create a session
5. create an htmlObjet(sesssion, iDservice URL and refresURL)
6. Sign and encrypt some portion of the htmlService with VIDP keys
1. BC makes opens a connection to IDService using the URL and Session returned during initAuthentication
2. End-User is presented with requested attributes for concent. If OK the attributes are gotten from the Security Token by the BC or MW.
3. The IDservice stores then information and after the getAuthentication these are destroyed
IDService
1. getAuthentication Data by providing the SessionID

Figure 12: Sequence Diagram - UC-AU-M-eIdService

Action	Description in the order of events
getResource	End User request for a resource
InitAuthentication	• SP initiates authentication process through VIDP-WS web services. • VIDP-WS does logging, authentication, and authorization and attribute validation. • Translate to STORKAuthRequest. • Calls VIDP with STORKAuthRequest. • VIDP decides to either to route to a C-PEPS or a SPWare. • SPWare for the specific country implementation does a mapping to member states' attributes and makes use of the SPWare table to make access to the country MW(e.g. eID-Service access with SAMLRequest signed and encrypted) • An HTT-Get/SAML request is sent to eID-Service • Updates session information with refreshUrl and sessionID from eID-Service • replaces the refreshUrl in httpobject from the eID-Service with the AssertionConsumerServiceUrl of the SP • returns the HttpObject to the SP which then returns as response to the browser
http-POST with refresh URL by BC Plugin	• The BC browser Plug-in makes an http-Post using the refreshURL in htmlObject to SP or S-PEPS • The SP calls the VIDP-WS getAuthenticationData while the S-PEPS will do simple http-POST(redirect to VIDP) • The VIDP-WS or the front end that receives request from S-PEPS logs, authenticates and authorizes SP/S-PEPS, and validates message. • VIDP-WS forwards call to VIDP. • The VIDP determines which SPWare or PEPSConnector to use. This is done through the use of service locator that returns either SPWare client of PEPSConnector. • If SPWare chosen and its Germany SPWare, it uses the sessionID from VIDP to retrieve the refreshURL using the SessionManager. • If authentication processing status still in pending mode, SPWare returns a PENDING Status response to VIDP, checks if polling is enabled globally or for this SP, it will keep polling for the results from eID-Service. Is VIDP allowed to store AuthenticationData after a successful polling? • If authentication is received, the SPWare does mapping to STORKAuthResponse and returns to VIDP. The VIDP returns it to VIDP-WS which then maps to eID-Service attributes and returns to SP, which returns the resource to the end user.

Table 67: General flow in a UC-AU-M-eIdService

B. Appendix Austria Integration

Austria's national electronic identification and authentication solution is also based on a user-centric approach that is called Austrian citizen card concept. The following sections briefly outline the Austrian eID architecture as well as its integration into the STORK middleware architecture and the general STORK concept.

B.1 Austrian eID architecture

The Austrian Citizen Card concept is primarily defined for secure identification and authentication of citizens at online governmental or business service providers. The according eGovernment act [11] describes a technological neutral approach for this concept, thus smart-cards, mobile devices or any other technological approach fulfilling the specification of the Austrian Citizen Card is applicable. In particular, the Austrian Citizen Card is used for the creation and verification of electronic signatures. Those digital signatures based on qualified certificates can be used for the verification of a citizen's authenticity in online proceedings.

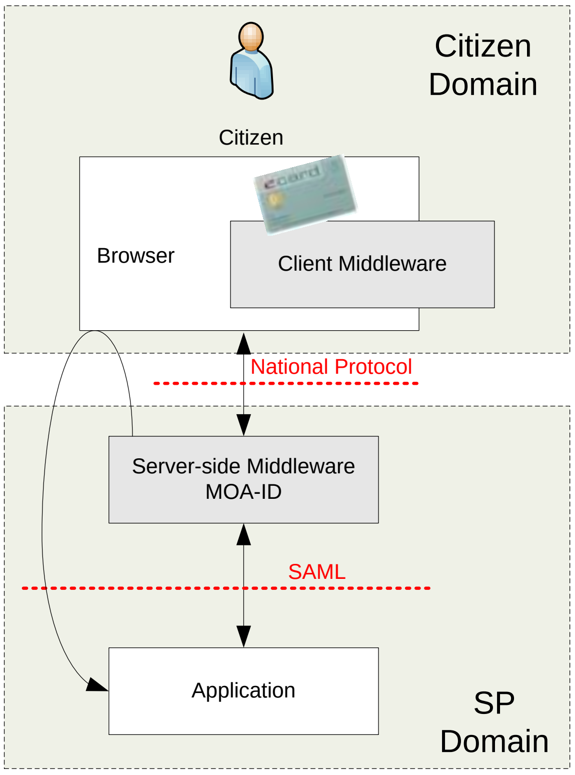


Figure 13: Austrian eID architecture

Figure 13 illustrates the middleware architecture applied in Austria based on smart-cards. The aim of this architecture is to decouple the actual identification and authentication process from the online application. The middleware actually consists of two parts, a client middleware and a server-side middleware. The client middleware handles the smart card communication while the server-side middleware manages the actual authentication process and the communication with applications of a service provider. The server-side middleware MOA-ID [7] has been developed to decouple a service provider from the card specifics. The client middleware allows the server-side middleware to access an Austrian eID card via a Web browser.

The client middleware can either be a piece of software running on the user's PC or a Java Applet running in the user's browser. In case of adopting the alternative using the Java Applet, this client middleware (called MOCCA [8]) is also divided into two parts. The Java Applet running on the citizen's client is responsible for the card-based communication with the Austrian eID cards. The client middleware running on a remote server executes computationally intensive operations needed for the communication between the server-side middleware MOA-ID and the eID cards.

In general, to enable citizens secure access to online services using their national eID, the service provider, e.g. a municipality, must at least run a server-side middleware MOA-ID and a server-based client middleware MOCCA if desired.

According to *Figure 13*, within the authentication architecture the following two important interfaces can be identified.

MOA-ID – Client Middleware:

Between the citizen's client middleware and MOA-ID a national protocol is used. This national interface is called Security Layer [9] and defines functions on an abstract level for the citizen card (e.g. creating digital signatures) which can be accessed by MOA-ID. The protocol used for communication between these two modules is based on XML. The XML-commands for the security layer can be bound to an arbitrary transport protocol such as TCP/IP or HTTP. In case of MOA-ID, HTTP over SSL/TLS is used.

MOA-ID – SP application:

MOA-ID provides a common and well-defined interface based on SAML (Security Assertion Markup Language) [10] for the exchange of authentication and identity information between MOA-ID and SAML-aware applications of a service provider. This message exchange protocol is based on the SAML Browser/Artifact Profile in version 1.0.

3.1 Process Flow

A process flow for an eID based authentication in Austria is shown in the sequence diagram in *Figure 14*.

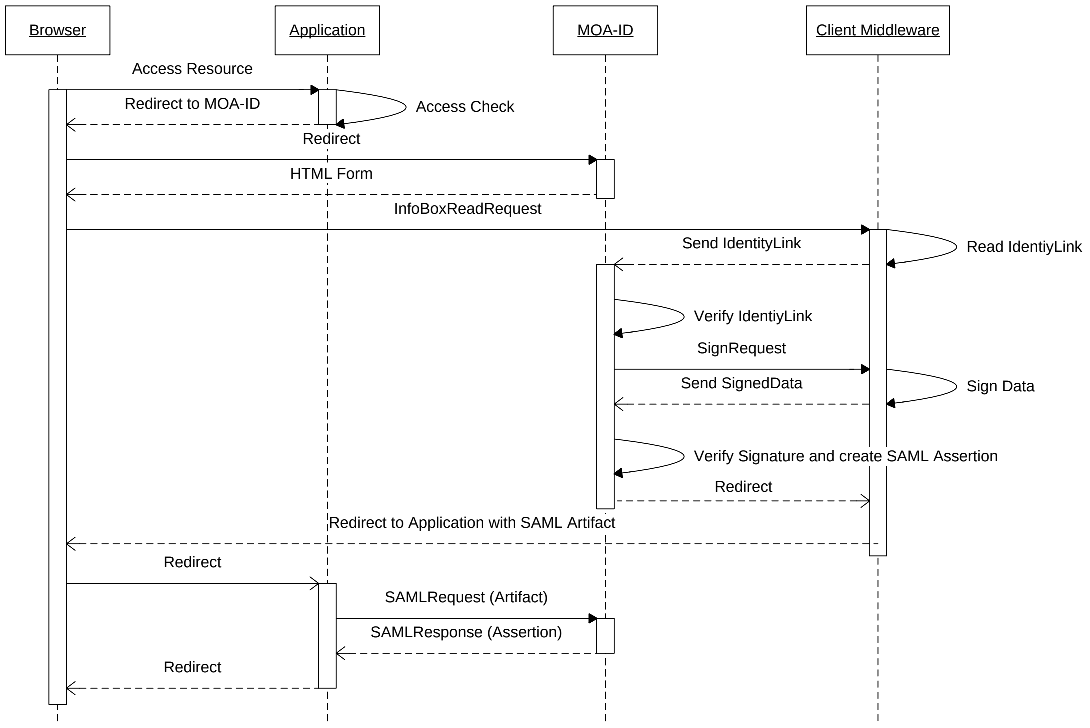


Figure 14: Authentication Process Flow in Austria

In this diagram, a user is requesting access to a citizen-card protected resource of a service provider. The application of the service provider checks whether a security context has already been established before. In this example the user is not yet authenticated and hence is redirected to the server-side middleware MOA-ID.

MOA-ID creates a session context and sends an XML-based, so-called InfoBoxReadRequest included in a HTML form to the client middleware via the user's browser. This request message is processed by reading a so-called identity link from the user's citizen card. The identity link defines a special data structure based on SAML containing identity information of the user. This comprises the user's unique identifier, the user's first and last name, and the user's date of birth. The identity link is sent back to MOA-ID and the attached signature is verified. After successful verification, MOA-ID sends a signature request to the client middleware containing data the user should sign for certifying the authentication appliance. The client middleware displays the user the data to be signed and transfers the result of the signature process back to MOA-ID again. MOA-ID verifies the applied signature and creates a SAML Assertion based on the user's identity information. Due to privacy regulations, the unique identifier stored on the citizen card must not be transferred to every application (e.g. not to business services). Therefore, the unique identifier is derived by a one-way hash function into a unique sector-specific identifier. The calculation of the sector-specific identifier depends on whether the respective application is a public or business service.

According to the SAML Browser/Artifact Profile, MOA-ID generates a so-called SAML artifact, which specifies a reference to the previously created SAML assertion. This artifact is appended to the URL of the actual requested application the user is redirected to afterwards. The artifact is used to dereference the SAML assertion. For that, the application assembles a SAMLRequest message including the artifact and sends it via back-channel communication (SOAP over HTTP(s)) to MOA-ID. Using the obtained artifact, MOA-ID looks up the corresponding assertion, wraps it into a SAMLResponse message and transfers it back to the requesting application. The application verifies the received identity and authentication information and grants or denies access to the requested protected resource.

B.2 Integration into STORK Middleware

This section describes the integration of the Austrian middleware concept into the STORK architecture. It can be distinguished between two different use cases where the Austrian middleware is involved. The first one addresses the use case where the Austrian middleware is triggered through the VIDP by a foreign S-PEPS (S-PEPS – VIDP – MOA-ID). The second use case defines the citizen authentication requested by an Austrian service provider (SP-AT – VIDP – MOA-ID).

B.2.1 Use Case: S-PEPS – VIDP – MOA-ID

This use case defines the authentication scenario for an Austrian citizen who wants to authenticate at a foreign service provider located in a so-called PEPS country. In this case, the actual authentication request at the service provider is forwarded to the PEPS which in turn calls the VIDP. The VIDP determines the appropriate SPWare and – for Austria – authentication is further processed by MOA-ID. The sequence diagram in *Figure 14* illustrates the individual process steps.

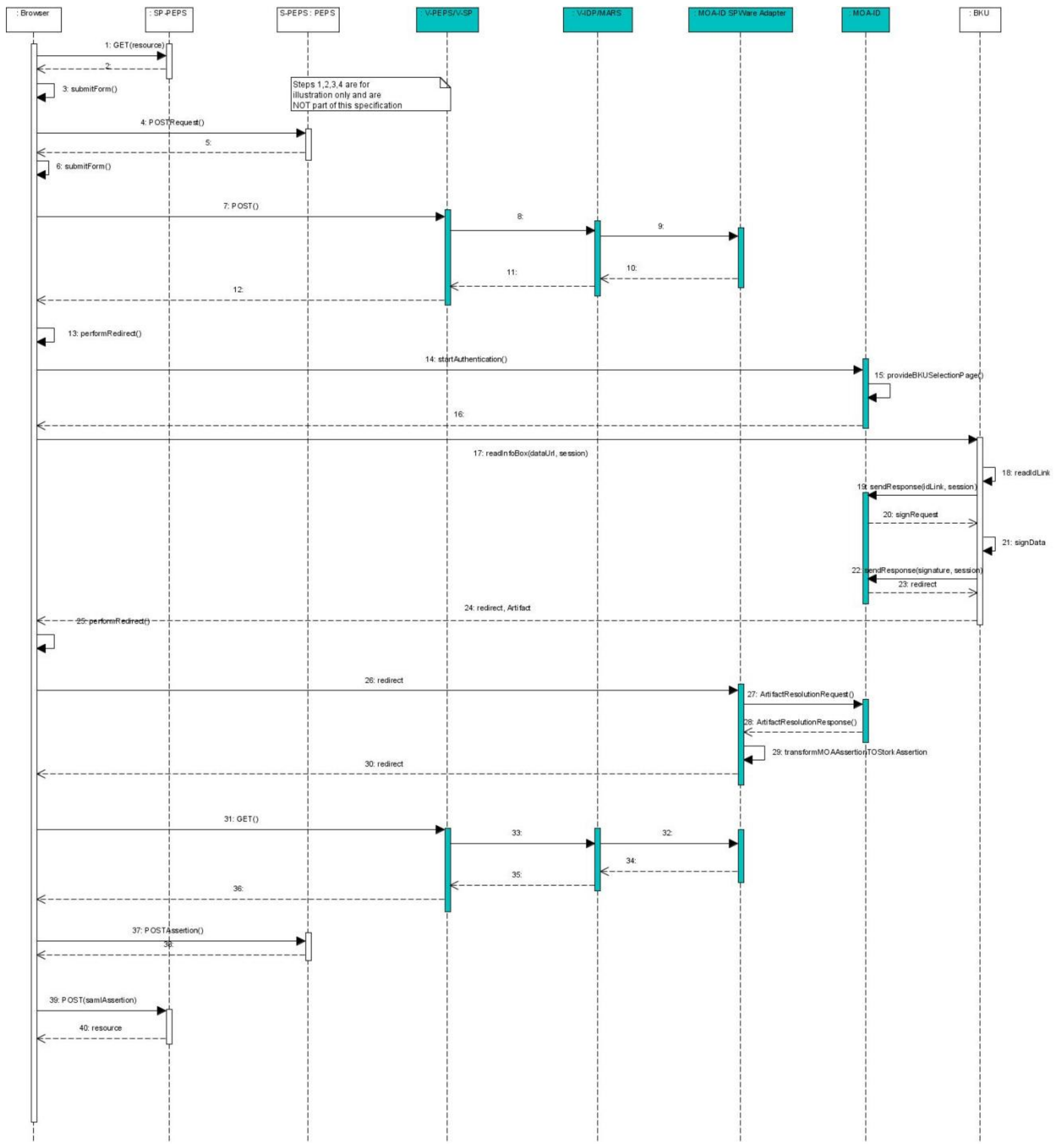


Figure 15: Austrian citizen to S-PEPS authentication

B.2.2 Use Case: SP-AT – VIDP – MOA-ID

In this use case a user wants to authenticate at an Austrian Service Provider. If the Austrian SP is not capable of the new SAML Web interface, legacy support is given. For this, a special adapter is developed that transfers legacy requests into STORK requests and vice versa. *Figure 16* illustrates the authentication flow for this use case.

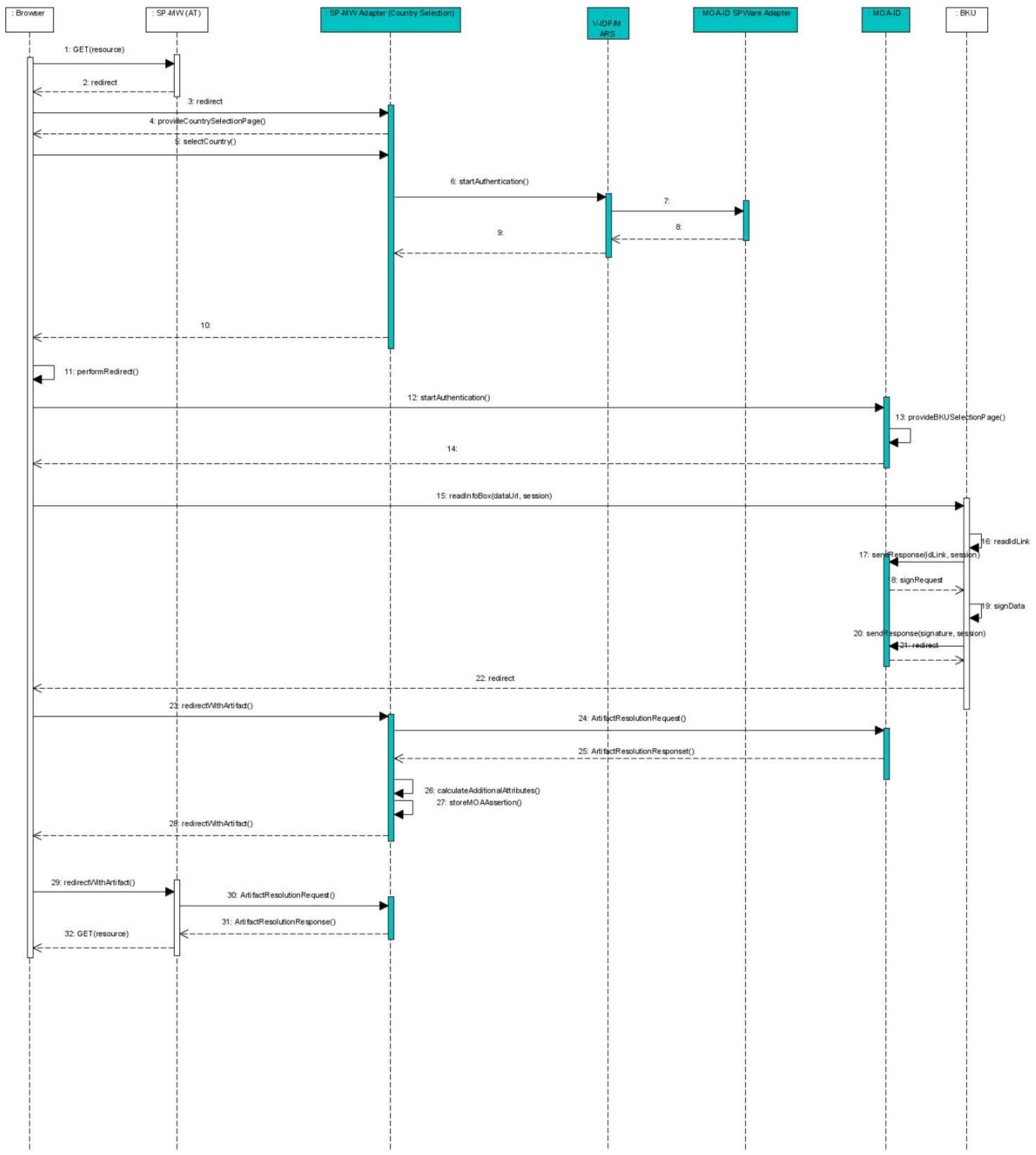

: Browser
: SP-MW (AT)
: SP-MW Adapter (Country Selection)
: V-IDP/MARS
: MOA-ID SPWare Adapter
: MOA-ID
: BKU
1: GET(resource)
2: redirect
3: redirect
4: provideCountrySelectionPage()
5: selectCountry()
6: startAuthentication()
7:
8:
9:
10:
11: performRedirect()
12: startAuthentication()
13: provideBKUSelectionPage()
14:
15: readInfoBox(dataUrl, session)
16: readIdLink
17: sendResponse(idLink, session)
18: signRequest
19: signData
20: sendResponse(signature, session)
21: redirect
22: redirect
23: redirectWithArtifact()
24: ArtifactResolutionRequest()
25: ArtifactResolutionResponset()
26: calculateAdditionalAttributes()
27: storeMOAAssertion()
28: redirectWithArtifact()
29: redirectWithArtifact()
30: ArtifactResolutionRequest()
31: ArtifactResolutionResponse()
32: GET(resource)

Figure 16: Austrian citizen to Austrian SP-MW authentication

C. Appendix C-PEPS integration

This section gives a brief summary about the integration of a C-PEPS into the STORK middleware architecture. It handles the use case where a citizen coming from a so-called PEPS country wants to authenticate at a service provider located in a middleware country. Details on this process flow have already been specified in deliverable D5.8.3a [3]. To keep this document self-contained, *Figure 17* illustrates such an authentication process again.

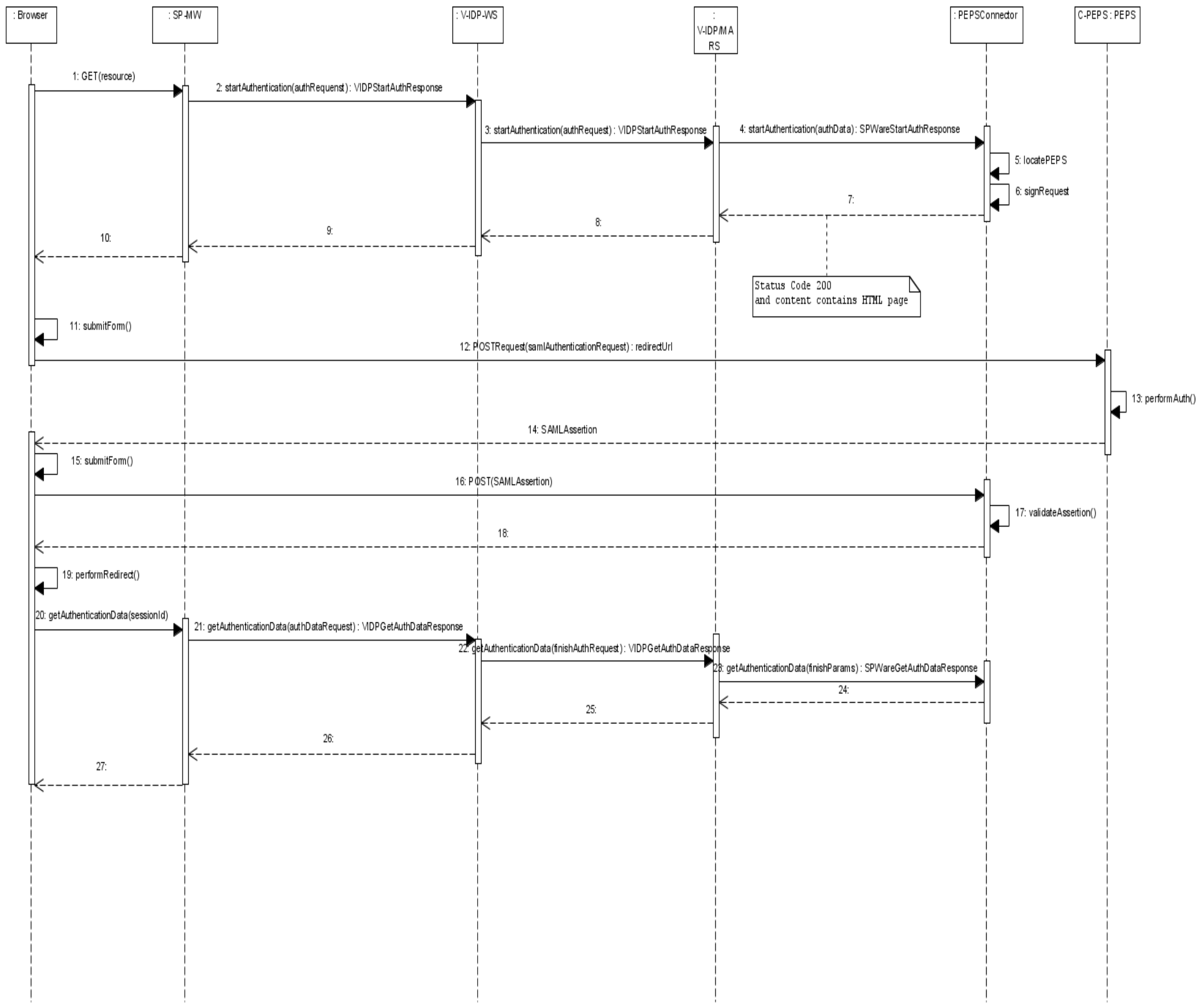

: Browser
: SP-MW
: V-IDP-WS
: V-IDP/MA RS
: PEPSConnector
C-PEPS : PEPS
1: GET(resource)
2: startAuthentication(authRequenst) : VIDPStartAuthResponse
3: startAuthentication(authRequest) : VIDPStartAuthResponse
4: startAuthentication(authData) : SPWareStartAuthResponse
5: locatePEPS
6: signRequest
7:
8:
9:
10:
Status Code 200
and content contains HTML page
11: submitForm()
12: POSTRequest(samlAuthenticationRequest) : redirectUrl
13: performAuth()
14: SAMLAssertion
15: submitForm()
16: POST(SAMLAssertion)
17: validateAssertion()
18:
19: performRedirect()
20: getAuthenticationData(sessionId)
21: getAuthenticationData(authDataRequest) : VIDPGetAuthDataResponse
22: getAuthenticationData(finishAuthRequest) : VIDPGetAuthDataResponse
23: getAuthenticationData(finishParams) : SPWareGetAuthDataResponse
24:
25:
26:
27:

Figure 17: Foreign PEPS citizen to SP-MW