WP51 – Citizen Services

D51.3 - Citizen Services Implementation

Document Identification	
Date	30/10/2015
Status	Final
Version	1.1

Related SP / WP	SP5 / WP51	**Document Reference**	D51.3
Related Deliverable(s)	D33.3, D41.4, D44.4, D51.2	**Dissemination Level**	PU
Lead Participant	FHG	**Lead Author**	Ben Kraufmann (FHG) Raik Kuhlisch (FHG)
Contributors	Florian Thiemer (FHG)	**Reviewers**	Jessica Schroers (KUL), Nuria Ituarte Aranda (ATOS)

Abstract: This document gives a description of the implementation details of WP51 and covers the integration of the FutureID means with the epSOS OpenNCP portal which acts as a service provider from the perspective of the FutureID reference architecture.

This document is issued within the frame and for the purpose of the *FutureID* project. This project has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement no. 318424.

1. Executive Summary

The FutureID project builds a comprehensive, flexible, privacy-aware and ubiquitously usable identity management infrastructure for Europe. To demonstrate and evaluate its deliverables, FutureID developed two pilot applications. One of them covers citizen services. Hence, epSOS – a project co-financed by the European Commission for a pan-European health data exchange cross border – was chosen to be extended with FutureID technology which reflects the overlap of FutureID and epSOS. An overlap exists for authenticating patients and digitally signing of documents. Four use cases complete existing epSOS functionalities:

- Digitally sign a patient privacy consent acknowledgment document in the country of treatment
- Authenticate patient via national certification authority means in the patient's country of affiliation
- Retrieve identity attributes locally in order to support a standardized demographics query for patient identification in the country of treatment
- Realize end-to-end security when disclosing medical data through establishing cryptographic keys based on short transaction numbers

This deliverable defines necessary implementation modules that accomplish these use cases.

2. Document Information

2.1 Contributors

Name	Partner
Florian Thiemer	FHG

2.2 History

Version	Date	Author	Changes
0.1	18/12/2014	Florian Thiemer	1st draft of the content list
1.0	27/10/2015	Raik Kuhlisch, Ben Kraufmann	Completed description of integration and implementation work.
1.1	30/10/2015	Raik Kuhlisch, Ben Kraufmann	Incorporated reviewer comments.

3. Table of Figures

4. Table of Tables

5. Table of Acronyms

AIS	**A**pplication **I**ntegration **S**ervice
ASN.1	**A**bstract **S**yntax **N**otation **O**ne
BPPC	**B**asic **P**atient **P**rivacy **C**onsents
CAS	**C**ustom **A**uthentication **S**ystem
CDA	**C**linical **D**ocument **A**rchitecture
DSI	**D**ata **S**tructure for **I**nteroperability
ECCF	**E**nterprise **C**onsistency and **C**onformity **F**ramework
EF	**E**lementary **F**ile
EHC	**E**lectronic **H**ealthcare **C**ard
eGK	**e**lektronische **G**esundheits**k**arte (EHC in Germany)
epSOS	**E**uropean **P**atients **S**mart **O**pen **S**ervices
ESS	**E**xtended **S**ecurity **S**afeguards
FAR	**F**utureID **A**uthentication **R**equest
HL7	**H**ealth **L**evel **S**even
HL7 CDA	**H**ealth **L**evel **S**even **C**linical **D**ocument **A**rchitecture
HTTP	**H**yper**t**ext **T**ransfer **P**rotocol
ISO	**I**nternational **O**rganization for **S**tandardization
JSON	**J**ava **S**cript **O**bject **N**otation
JWT	**J**SON **W**eb **T**oken
LARMS	**L**ocal **A**ttribute **R**etrieval and **M**apping **S**ervice
LAS	**L**ocal **A**uthentication **S**ervice
NCP	**N**ational **C**ontact **P**oint
OASIS	**O**rganization for the **A**dvancement of **S**tructure **I**nformation
OASIS DSS	**O**rganization for the **A**dvancement of **S**tructure **I**nformation Digital Signature Service

PAKE	**P**assword **A**uthenticated **K**ey **E**xchange
SAIF	**S**ervice-**A**ware **I**nteroperability **F**ramework
SAML	**S**ecurity **A**ssertion **M**arkup **L**anguage
STORK	**S**ecure Iden**t**ity Acr**o**ss Bo**r**ders Lin**k**ed
TAN	**T**rans**a**ction **N**umber
URI	**U**niform **R**esource **I**dentifier
XML	E**x**tensible **M**arkup **L**anguage

6. Referenced Documents

[1] OpenNCP Community, 2014. [Online]. Available: https://openncp.atlassian.net/wiki/display/ncp/OpenNCP+Community+Home.

[2] FutureID Consortium, *WP 51 – Citizen Services. D51.2 – Technical Module and Interface Specification. Version 1.2,* 2014.

[3] Health Level Seven, "HL7 Service Aware Interoperability Framework (SAIF)," 2011. [Online]. Available: http://hssp.wikispaces.com/HL7+SAIF.

[4] Health Level Seven International, *HL7 Implementation Guide for CDA Release 2: Privacy Consent Directives, Release 1,* May 2014.

[5] FutureID Consortium, *WP 33 – eSign Services. D33.2 – Interface and Module Specification and Documentation. Version 0.6,* 2013.

[6] J. Caumanns, S. Bittins, D. Hühnlein and T. Wich, "epSOS Extended Security Safeguards (ESS): AP-03 – epSOS ESS Specification. Version 1.1," 2012.

[7] F. Hao and P. Ryan, "Password Authenticated Key Exchange by Juggling," in *Security Protocols XVI*, vol. 6615, Springer Berlin Heidelberg, 2011, pp. 159-171.

7. Table of Contents

8. Project Description

The FutureID project builds a comprehensive, flexible, privacy-aware and ubiquitously usable identity management infrastructure for Europe, which integrates existing eID technology and trust infrastructures, emerging federated identity management services and modern credential technologies to provide a user-centric system for the trustworthy and accountable management of identity claims.

The FutureID infrastructure will provide great benefits to all stakeholders involved in the eID value chain. Users will benefit from the availability of a ubiquitously usable open source eID client that is capable of running on arbitrary desktop PCs, tablets and modern smart phones. FutureID will allow application and service providers to easily integrate their existing services with the FutureID infrastructure, providing them with the benefits from the strong security offered by eIDs without requiring them to make substantial investments.

This will enable service providers to offer this technology to users as an alternative to username/password based systems, providing them with a choice for a more trustworthy, usable and innovative technology. For existing and emerging trust service providers and card issuers, FutureID will provide an integrative framework, which eases using their authentication and signature related products across Europe and beyond.

To demonstrate the applicability of the developed technologies and the feasibility of the overall approach FutureID will develop two pilot applications and is open for additional application services who want to use the innovative FutureID technology

FutureID is a three-year duration project funded by the European Commission *Seventh Framework Programme* (FP7/2007-2013) under grant agreement no. 318424

9. Introduction

9.1 Scope

This document defines how the FutureID components are integrated and used with the Community Portal *OpenNCP* which has been developed by the Open National Contact Point (OpenNCP) Community [1].

9.2 On OpenNCP

The OpenNCP portal is a Java-based implementation of the epSOS services with a Web-based user interface conforming to the Java Portlet Specification 2.0 (JSR 286). The pan-European Open Source health information exchange ecosystem of the OpenNCP provides both an implementation of a National Contact Point in the role of a data requestor (short 'OpenNCP-B') and a National Contact Point in the role of a data provider (short 'OpenNCP-A).

FutureID is integrated in order to enhance OpenNCP-B means. Thus, the authentication of physicians as well as signing patient privacy consents with an existing eID/smartcard has been supported. The Figures 1 and 2 give an impression of the Web-based user interface.

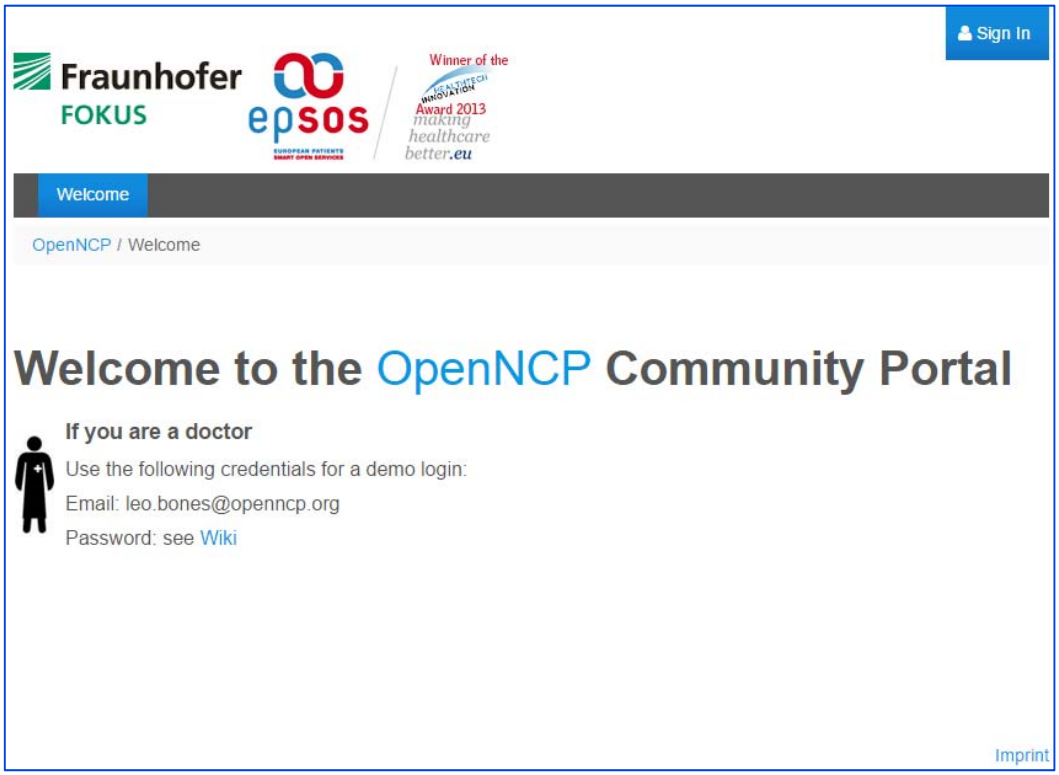

Figure 1: Front page of the OpenNCP Portal

Figure 2: OpenNCP's Country Selection Mask: Starting point for pan-European health-information exchange

9.3 Approach

This document proceeds with the approach already defined in D51.2 [2]. Thus, the Health Level Seven's (HL7) *Service-Aware Interoperability Framework* (SAIF) [3] approach has been applied.

The following table displays all named dimensions and the related sections in this document.

Table 1: Enterprise Consistency and Conformity Framework (ECCF) specification scope and the related sections in D51.3

Specification Scope	**Enterprise Dimension** *"Why" Policy*	**Information Dimension** *"What" Content*	**Computational Dimension** *"How" Behaviour*	**Engineering Dimension** *"Where" Implementation*	**Technical Dimension** *"Where" Deployment*
Conceptual Perspective	n/a	n/a	n/a	n/a	n/a
Logical Perspective	n/a	n/a	n/a	n/a	n/a
Implementable Perspective	n/a	n/a	n/a	Section 10	n/a

Hardware platforms and execution environments are out of scope of this specification. That is why explanations with regard to the technical dimension are not given in this document

10. Implementable Specification

This perspective of the engineering dimension includes all necessary technical linkages (e.g., data types, value sets, interface specifications etc.) and allows developers to implement components of the functional/logical specification using standards-based technology components.

10.1 Signature Service

The FutureID Client comes with an optional signature plugin that is able to digitally sign XML documents as well as PDF documents. Due to the fact that the signature plugin has an HTTP binding based on the OASIS Digital Signature Services (DSS) specification, documents can be sent to a dedicated URL for a signature request.

Each disclosure of medical data in the OpenNCP-B portal requires submitting a patient privacy consent acknowledgment document [4] to OpenNCP-A which in turn checks the authorized return of patient data in the country of affiliation. Before the submission of that form, a digital signature is requested.

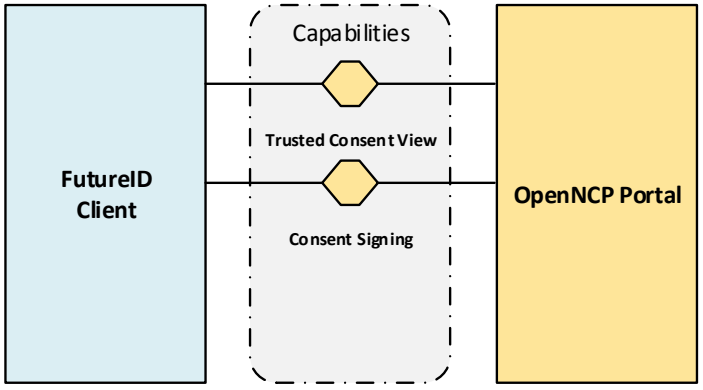

Figure 3: Signature Service Capabilities

The signature service capabilities are served by the eSign plugin [5]. Readers are directed to this deliverable for further implementation details. The OpenNCP invokes this plugin through sending a signature request conforming to OASIS DSS that holds the patient privacy consent acknowledgment document in XML format to the Localhost interface accessible via https://www.localhost-ecard-client.de:24728/eSign. The invocation is done via JavaScript as it is shown in Figure 4.

```
function extractSignedDoc(dssResponse) {
```

```
    var b64 = dssResponse.evaluate( "//*[local-name()='Base64XML']/text()",
dssResponse, null, XPathResult.STRING_TYPE, null )
    console.log(b64);
    if (b64) {
        var signedDoc = decodeURIComponent(escape(window.atob(
b64.stringValue )));
        return signedDoc;
    }
}

function signConsent(){
      var consent =
      document.getElementById('_portalb_WAR_epsosportal_:consentForm:consent
      ').value;

      var request = "<dss:SignRequest> … <!—DSS Template: see Appendix B-->"
      request = request + consent;
      request = request + "… </dss:SignRequest>";

      $.ajax({
        type: 'POST',
        url: 'https://www.localhost-ecard-client.de:24728/eSign',
        crossDomain: true,
        contentType: 'text/plain',
        async: false,
        data: request,
        success: function(responseData) {
            var doc = extractSignedDoc(responseData);
            document.getElementById('_portalb_WAR_epsosportal_:consentForm:s
            ignedConsent').value = doc;
        } ,
        error: function (responseData, textStatus, errorThrown) {
            console.log("failed to sign");
        }
      });
      return false;

};
```

Figure 4: JavaScript to invoke the eSign plugin locally

This function is integrated with the OpenNCP portal.

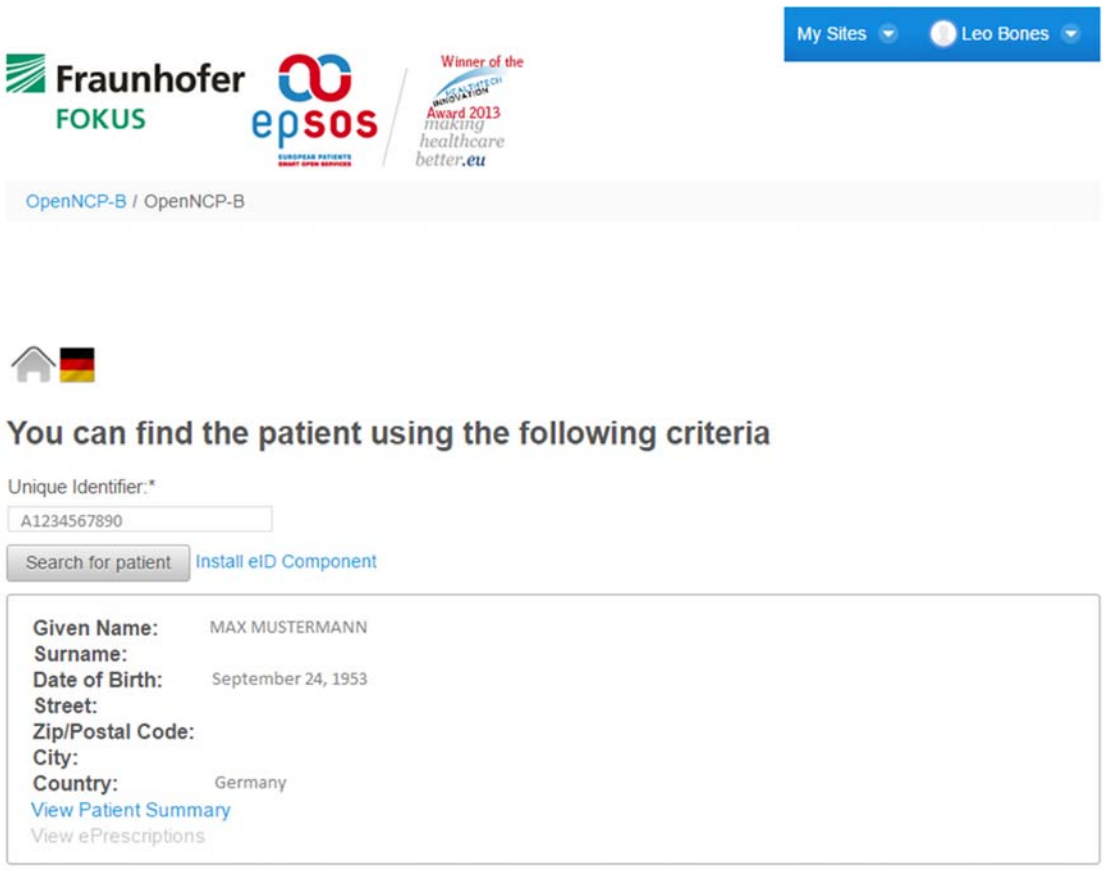

Figure 5: Find a patient with specific demographics

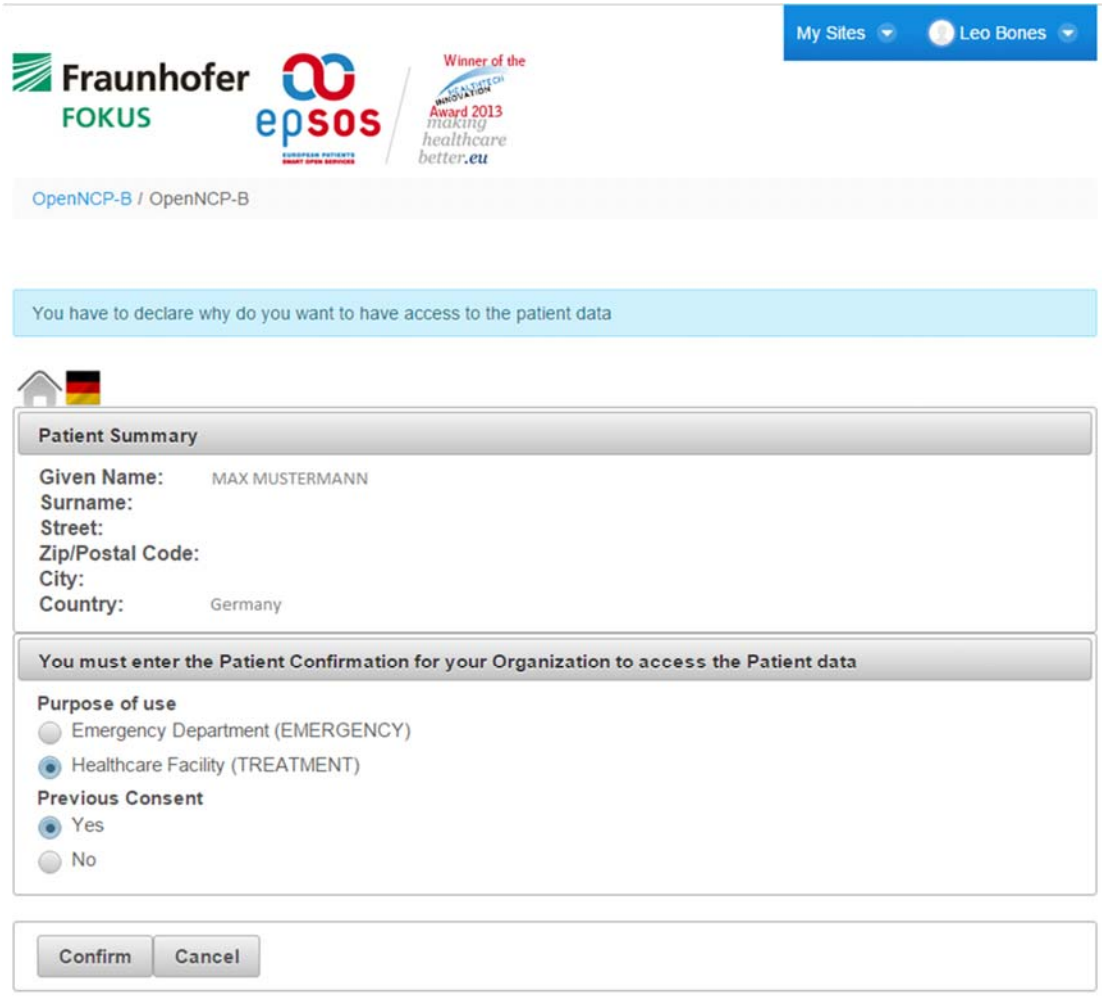

Figure 6: Declaration of patient confirmation

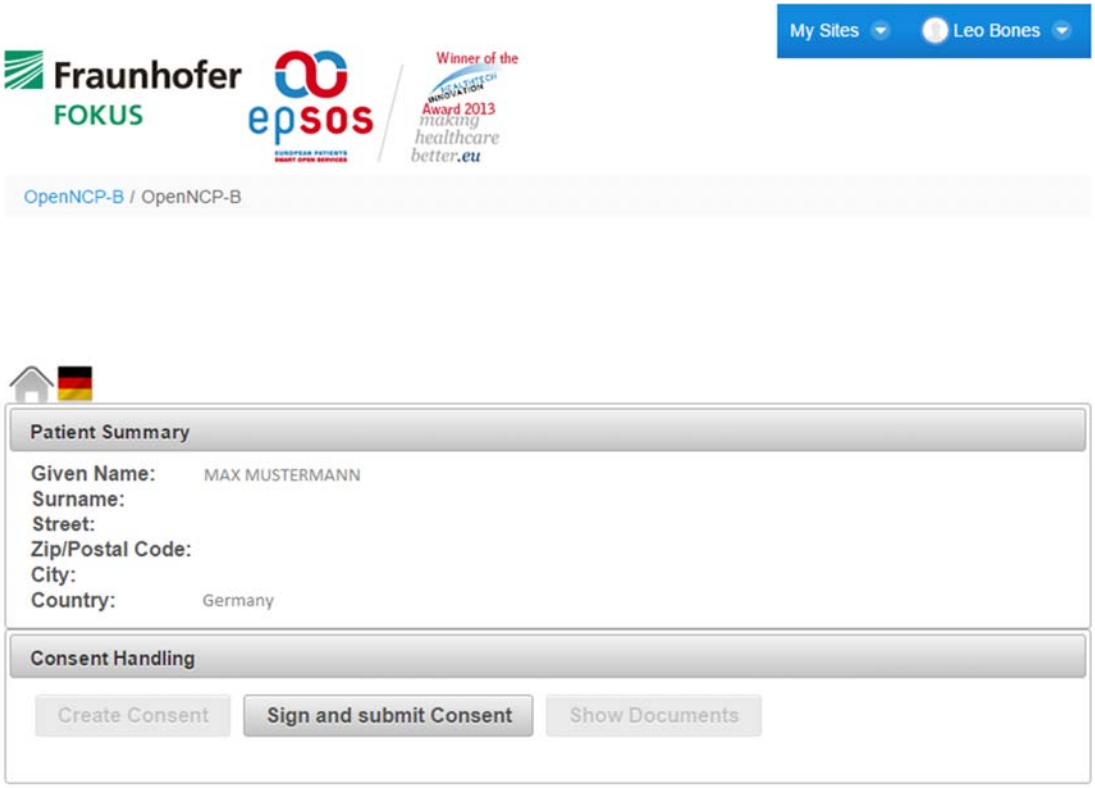

Figure 7: Trigger consent signature

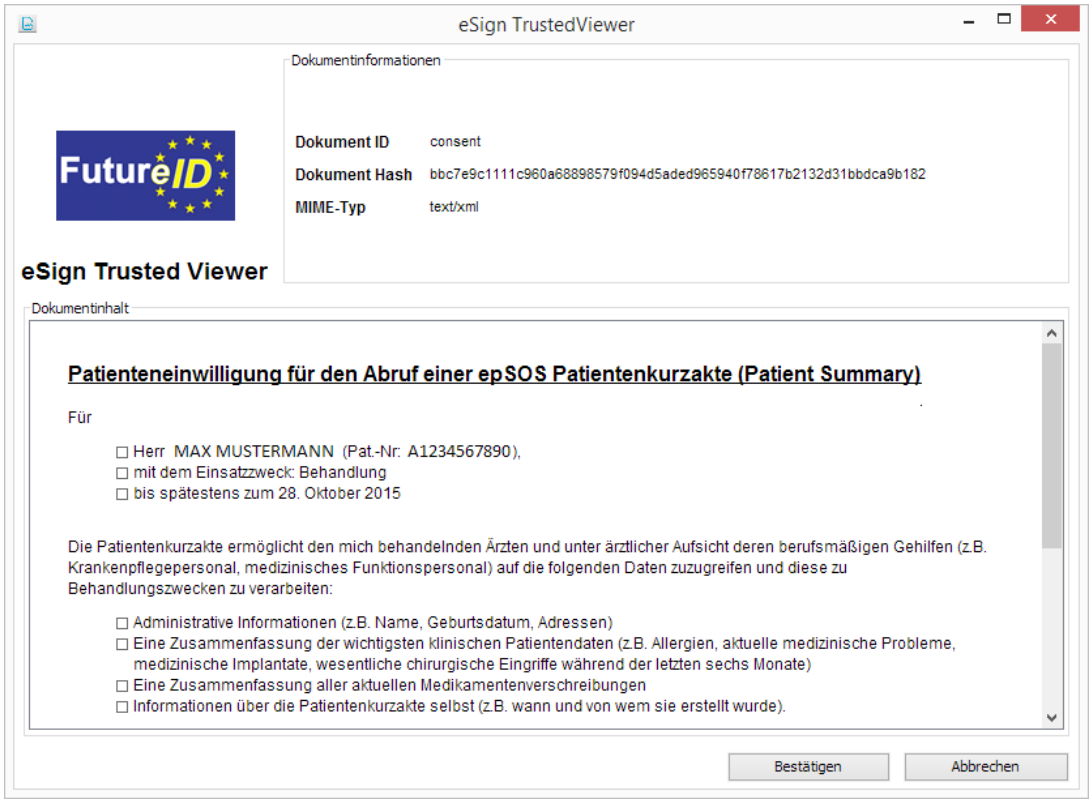

Figure 8: Consent, displayed in Trusted Viewer

10.2 Local Authentication Service

The OpenNCP portal is implemented with the Liferay Portal Community Edition 6.2 CE GA2[1] deployed on the application server Apache Tomcat/7.0.47. Liferay supports various extensions via hooks. The Auto Login Hook[2] allows to inspect incoming HTTP requests and do custom authentication token validation and to perform a login automatically. The Local Authentication Service is implemented by this hook. The hook is called when either a user has not be authenticated or the previous session has timed out.

Registered portal users are associated with a pseudonymised eIdentifier from the FutureID Broker by the local administrator. This enables the filter class of the hook to retrieve a valid eID/user mapping from the local identity store.

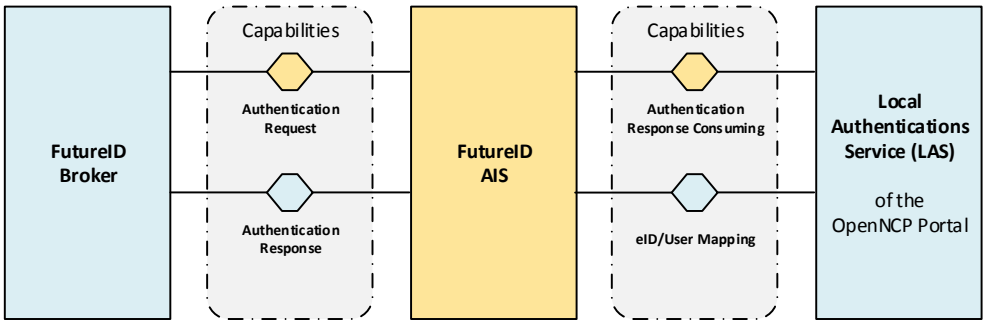

Figure 9: Local Authentication Service Capabilities

The implementation of Auto Login Hook is given in Appendix C.

A user may login using the FutureID capability the same way as she would sign in using the common username-password pattern. The integration of FutureID with OpenNCP comprised the inclusion of a new Sign In button (cf. Figure 9). This button triggers the FutureID authentication process. The Web browser redirects to the FutureID Solver and Executor component as a sub-component of the FutureID Broker (cf. Figure 11). As a result, the Auto Login Hook will receive a SAML Response message that includes the attribute eIdentifier.

[1] See https://www.liferay.com/.
[2] See http://www.liferay.com/community/wiki/-/wiki/Main/Developing+a+CAS#section-Developing+a+CAS-AutoLogin.

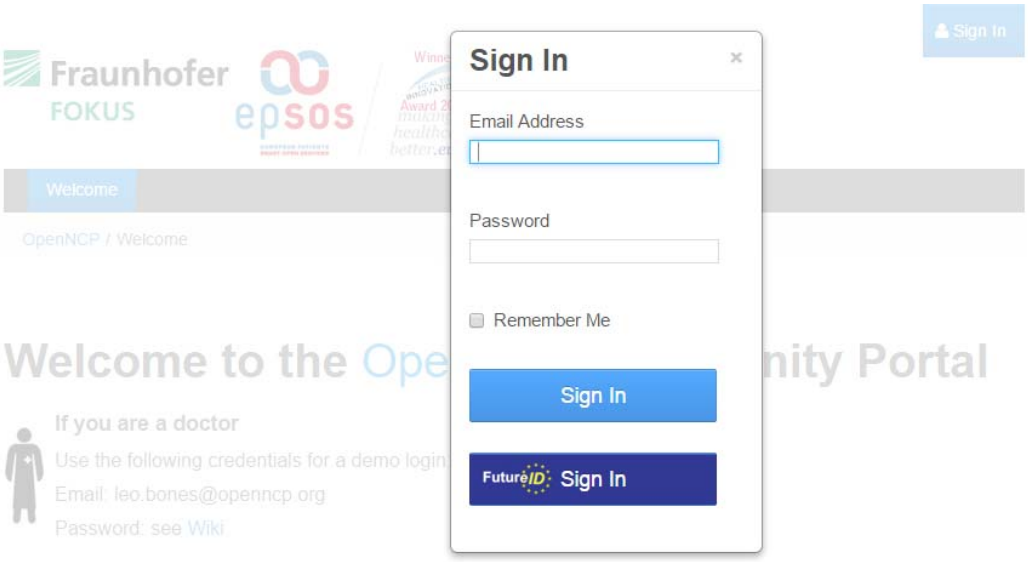

Figure 10: Use FutureID Login capability on OpenNCP Portal Web page

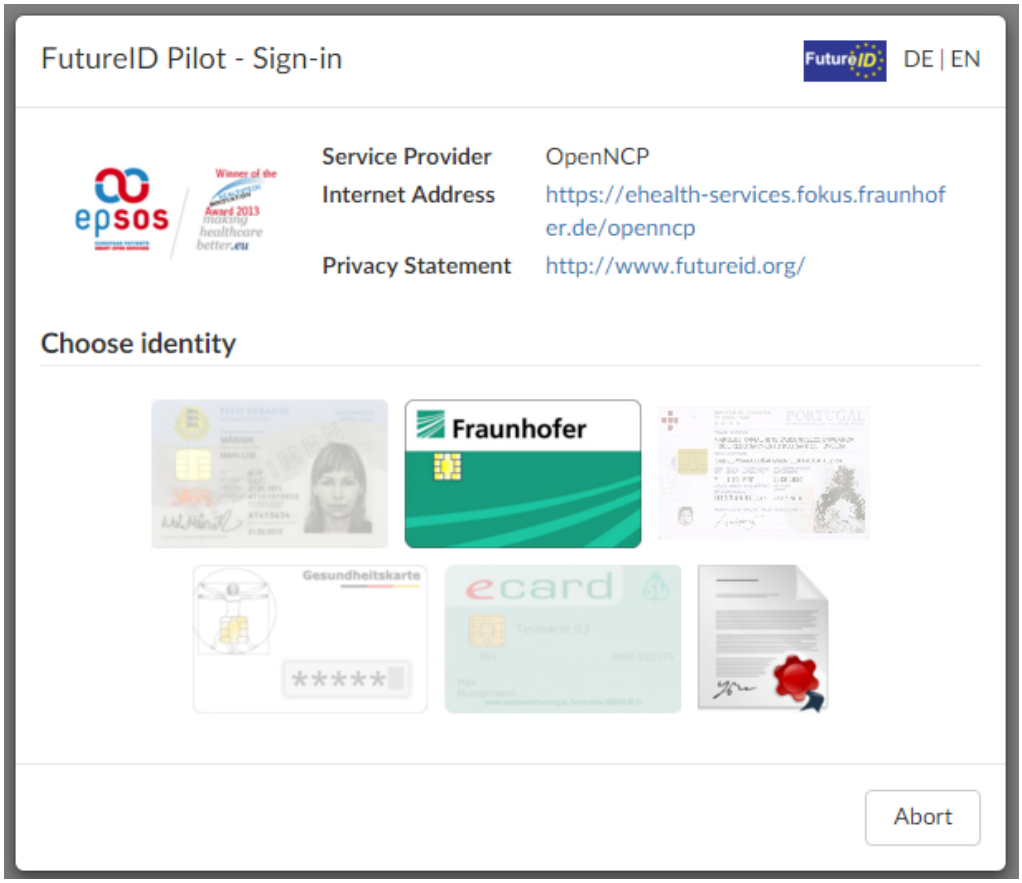

Figure 11: FutureID Solver and Executor

10.3 Local Attribute Retrieval and Mapping Service

The FutureID Client is used as a framework for the Local Attribute and Mapping Service (LARMS). The FutureID Client has an ISO 24727 compatible library for communication operations with a smartcard. LARMS is implemented as a FutureID plugin.

The plugin has the following structure:

```
larms.jar
|
|---your
|   |---package
|       |---name
|           |---ExtendedPluginAction.java
|---META-INF
    |---Addon.xml
```

The plugin comprises:

1. A `META-INF/Addon.xml` file,
2. The implementation class that serves as an entry point to the plugin.

The Addon.xml file links an HTTP endpoint to a resource name. If the FutureID client receives a request, it extracts the resource name and looks for a corresponding handler class in order to process the request.

The `Addon.xml` file contains the following configuration properties:

Table 2: Addon Specification

Element	Description	Example Value
AddonSpecification		
ID	A unique id for each plugin	LARMS
Version	A version for this plugin	`0.1`
BindingActions		
AppPluginSpecification		
ClassName	The fully qualified class name for the Plugin Entry Point class.	`org.openecard.plugins.larms.ExtendedPluginAction`
ResourceName	Resource path linked to the Plugin Entry Point class	`larms/getAttributes`

11. An example `Addon.xml` file is given in Conclusion

The implementation of the Citizen Services Pilot comprises the extension of the OpenNCP portal in order to allow for external authentications as well as eID-based signing. Enabling OpenNCP to use FutureID as an authentication schema was easy, as interfaces rely on well-known and mature standards. The tricky part was to enable OpenNCP to use external authentication schemes in general. This pilot implementation revealed that existing epSOS processes such as the consent-based process could be improved due to an enhanced patient involvement (i. e. eID-based signature).

Appendix A – LARMS Plugin Addon.xml File

The implementation of the endpoint is in the file `ExtendedPluginAction.java`. The class implements the `AppPluginAction` interface. The FutureID client provides the implementation with the HTTP request message that may trigger the plugin. The implementation handles the request and returns an object of type `BindingResult`.

```
package your.package.name;

import java.util.Map;
import java.util.List;

import org.openecard.addon.Context;
import org.openecard.addon.bind.AppPluginAction;
import org.openecard.addon.bind.Attachment;
import org.openecard.addon.bind.BindingResult;
import org.openecard.addon.bind.Body;

/**
 * Main class of the Plugin. It gets called when a request for a resource
 * this plugin can handle arrives through a binding. The internal processing
 * mainly takes place according to the requests action parameter.
*/
public class ExtendedPluginAction implements AppPluginAction {
    private Context context;

    @Override
    public void init(Context ctx) {
        this.context = ctx;
    }

    @Override
    public void destroy() {
        // nothing to do
```

```
    }

    /**
     * Entry point method, when a HTTP request arrives.
     * @param requestBody content of the HTTP request
     * @param parameters GET parameter map
     * @param attachments
     * @return
     */
    @Override
    public BindingResult execute(Body requestBody,
        Map<String, String> parameters, List<Attachment> attachments) {
        /* Your Implementation goes here */
        BindingResult result = ...;
        return result;
    }
}
```

An overview of the LARMS plugin classes, configuration and the required FutureID Client functionality is given in Figure 12.

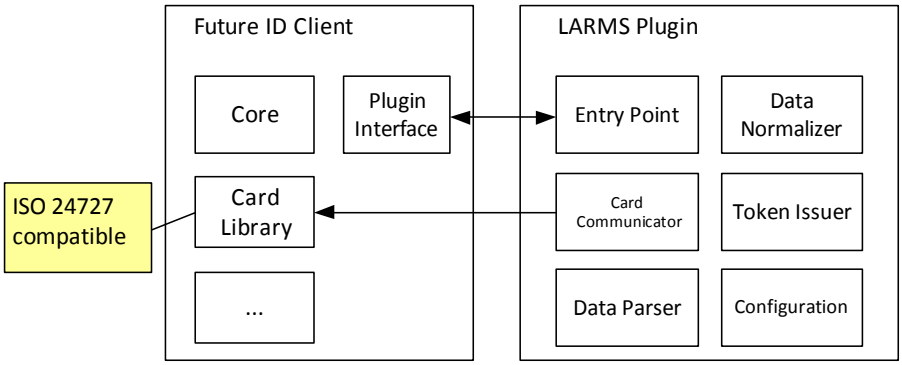

Figure 12: Technical component view of the LARMS plugin

The LARMS plugin has the following six core classes (see Figure 13):

- Entry Point
- Card Communicator
- Data Parser
- Data Normalizer
- Token Issuer
- Configuration

The class **Entry Point** is called from the FutureID Client's plugin interface implementation. This FutureID Client plugin is implemented as a HTTP endpoint. The FutureID Client passes

incoming HTTP requests dedicated to the LARMS plugin to the Entry Point implementation. Thus, the Entry Point parses the request, processes it according to the LARMS specification, constructs an appropriate result message and returns the result message to the FutureID Client plugin interface implementation.

The **Card Communicator** implements card identification and data reading on a higher level. The data read from a smartcard is received as a byte array.

The **Data Parser** transforms the byte array into an appropriate intermediate format for further data normalization processing. Intermediate format may be XML, ASN1, DER encoded data, etc. For each supported smartcard type, the implementation of the Data Parser has to transform the byte array representation in a proper format.

The **Data Normalizer** takes the identity attributes provided by the Data Parser and maps them to the predefined format that is a map of key-value pairs.

The **Token Issuer** constructs a result token depending on the requested format at the Entry Point. Valid formats are SAML Assertion and JWT. The token contains the identity attributes provided by the Data Normalizer.

The **Configuration** is a Java façade to the configuration source files.

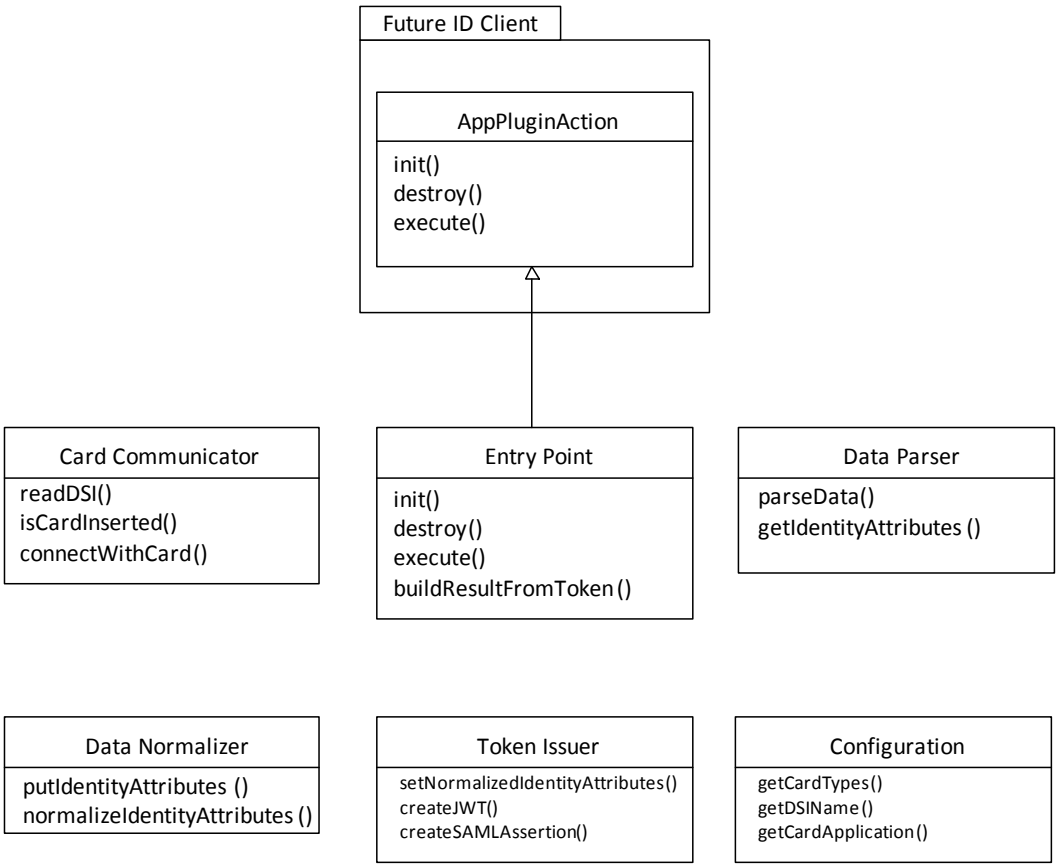

Figure 13: Class diagram of the LARMS plugin classes.

The sequence flow controlled by the class Entry Point is described in Figure 14.

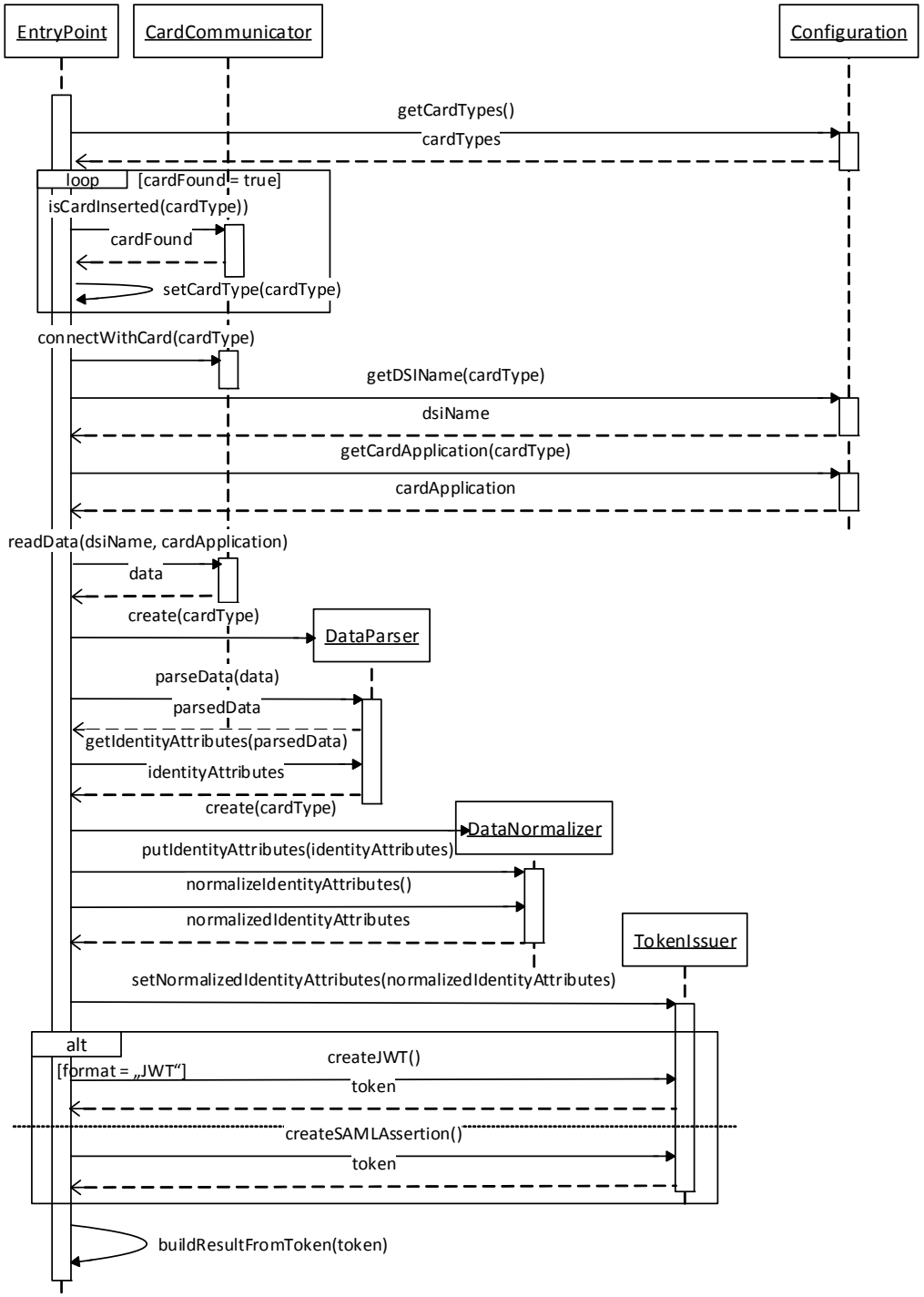

Figure 14: Sequence diagram for the LARMS plugin

The **Entry Point** retrieves the set of supported card types. The FutureID Client does not support an operation to estimate the current inserted card type. The only way is to ask the FutureID smartcard framework whether the inserted card is of a certain type. Thus, the Entry Point enters a loop for each supported card type and asks if the smartcard has the provided card type. If so, the loop finishes and the Entry Point establishes a connection with the smartcard.

The Entry Point queries the name of the DSI (Data Structure for Interoperability) and the application name. The Entry Point is able to read the byte array that contains the user's identity information.

After the `readDSI()` operation, the Entry Point instantiates the **Data Parser** with the card type. For each card type, the Data Parser triggers a distinct implementation. Data streams provided by a distinct card type follow a distinct schema. Afterwards, the Entry Point uses the Data Parser to extract the identity attributes and stores them in a map.

The **Data Normalizer** receives the identity attributes from the Entry Points and maps them into the predefined format.

Depending on the requested format, the Entry Point invokes either `createJWT()` or `createSAMLAssertion()`. Finally, a subsequent call to `buildResultFromToken()`creates the response message.

11.1.1 CardCommunicator Specification

The FutureID client provides an implementation to read data and meta-information from a smartcard. To simplify the access to a smartcard a CardCommunicator class is provided with the following methods:

- readDSI
- isCardInserted
- connectWithCard

Table 3: CardCommunicator - readDSI method specification

Method Name	readDSI		
Description	Read the contents of the DSI identified through `dsiName` and `cardApplication` and return the contents.		
Params	String	dsiName	Name of the DSI. For example the String `EF.PD` represents the personal information of the

			card holder of an eGK.
	byte[]	cardApplication	Identifier of the card application. For example the byte representation of the String `D040000017010101` representing the identifier for the card application of an eGK.
Returns	byte[]	Byte array representation of the requested DSI.	
Exceptions	org.openecard.common.WSHelper.WSException		
	org.openecard.common.interfaces.DispatcherException		
	java.lang.reflect.InvocationTargetException		

Table 4: CardCommunicator - isCardInserted method specification

Method Name	isCardInserted		
Description	Just gives the information, if a card of the given type is inserted or not.		
Params	String	cardType	Type of the card (e.g. http://ws.gematik.de/egk/1.0.0 for the German healthcare card)
Returns	Boolean	True, if a card is inserted.	
Exceptions	-		

Table 5: CardCommunicator - connectWithCard method specification

Method Name	connectWithCard		
Description	Initiates a connection with the specified card type.		
Params	String	cardType	Type of the card (e.g. http://ws.gematik.de/egk/1.0.0 for the German healthcare card)
Returns	-		
Exceptions	`org.openecard.plugins.larms.chain.helper.CardCommunicator.CardNotInsertedException`		

Table 6: CardCommunicator - connectWithCard method specification

11.1.2 DataParser Specification

Table 7: DataParser - parseData method specification

Method Name	parseData		
Description	This method parses the provided byte array in an appropriated data format representation. It is up to the implementer to choose a suitable representation.		
Params	Byte[]	Data	The byte array of the data to parse.
Returns	T (generic type)		An object of the generic type T representing the provided data.
Exceptions	-		

Table 8: DataParser - getIdentityAttributes method specification

Method Name	getIdentityAttributes
Description	This method gets the identity attributes from the provided parsed data object.

Params	T (generic type)	parsedData	The parsed data representation
Returns	Map<String, String>	The identity attributes as a hash map.	
Exceptions	-		

11.1.3 DataNormalizer Specification

Table 9: DataNormalizer - putIdentityAttributes method specification

Method Name	putIdentityAttributes		
Description	This methods sets the identity attributes.		
Params	Map<String, String>	identityAttributes	The identity attributes to set.
Returns	-		
Exceptions	-		

Table 10: DataNormalizer - normalizeIdentityAttributes method specification

Method Name	normalizeIdentityAttributes	
Description	This method normalizes the identity attributes.	
Params	-	
Returns	Map<String, String>	A hash map containing the normalized identity attributes.
Exceptions	-	

11.1.4 TokenIssuer Specification

Table 11: TokenIssuer - setNormalizedIdentityAttributes method specification

Method Name	setNormalizedIdentityAttributes
Description	This method sets the normalized identity attributes.

Params	Map<String, String>	A hash map containing the normalized identity attributes.
Returns	-	
Exceptions	-	

Table 12: TokenIssuer - createJWT method specification

Method Name	createJWT	
Description	This method creates the JWT from the normalized identity attributes.	
Params	-	
Returns	String	A string representation of the JWT.
Exceptions	-	

Table 13: TokenIssuer - createSAMLAssertion method specification

Method Name	createSAMLAssertion	
Description	This method creates the SAML Assertion from the normalized identity attributes.	
Params	-	
Returns	org.opensaml.saml2.core.Assertion	The SAML Assertion created from the normalized identity attributes.
Exceptions	-	

11.1.5 Configuration Specification

The configuration file of the LARMS plugin consists of an array of card type configurations. Each card type configuration has a country, card type, dsi name and card application property. A json schema that describes the configuration file's property and content is described in Figure 15.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Card Type Configuration set",
```

```
    "type": "array",
    "items": {
        "title": "Card Type Configuration",
        "type": "object",
        "properties": {
            "country": {
                "description": "The country code for this specific card
type.",
                "type": "string"
            },
            "CardType": {
                "description": "The unique identifier for a specific card
type.",
                "type": "string"
            },
            "DSIName": {
                "description": "The name of the DSI where the identity
information of the card holder is stored. This DSIName is associated with
the CardType property in this object.",
                "type": "string"
            },
            "CardApplication": {
                "description": "The name of the card application. This
CardApplication is associated with the CardType property in this object.",
                "type": "string"
            }
        },
        "required": ["country", "CardType", "DSIName", "CardApplication"]
    }
}
```

Figure 15: Data format specification for the LARMS plugin's configuration file

The following helper methods needs to be implemented in order to access the configuration file content:

- getCardTypes
- getDSIName
- getCardApplication

Table 14: Configuration - getCardTypes method specification

Method Name	getCardTypes
Description	This method returns all card type values that are present in the configuration file.
Params	-

Returns	List<String>	A list of card type values.
Exceptions	-	

Table 15: Configuration - getDSIName method specification

Method Name	getDSIName		
Description	This method returns the DSI name that is associated with the given card type.		
Params	String	cardType	The card type.
Returns	String		A DSI name that is associated with the given card type.
Exceptions	-		

Table 16: Configuration - getCardApplication method specification

Method Name	getCardApplication		
Description	This method returns the card application that is associated with the given card type.		
Params	String	cardType	The card type.
Returns	String		A card application that is associated with the given card type.
Exceptions	-		

Table 17: Configuration content of the LARMS plugin implementation

Country	CardType	DSIName	CardApplication
DE	http://ws.gematik.de/egk/1.0.0	EF.PD	D27600000102
BE	http://eid.belgium.be/eid	ID-RN	A000000177504B43532D3135
AT	http://cif.chipkarte.at/e-card/g3	Grunddaten	D040000017010101
IT	http://www.progettocns.it/cns	NKAF	a000000073

11.1.6 Integration with OpenNCP

The LARMS plugin is triggered by an interaction component on the OpenNCP Web page. It is important to note, that not the portal's backend triggers the plugin. Instead it is the browser, running locally with the FutureID Client. The interaction component may be a link or a button that triggers the JavaScript function `readAttributes()`. The function consists of:

1. Call the LARMS plugin,
2. Parse the response,
3. Fill in a form with smartcard attributes automatically.

Calls to the LARMS plugin are achieved with the XMLHttpRequest object. For simplified access the jQuery ajax library can be used:

```
function readAttributes() {

    $.ajax({
      url: 'https://www.localhost-ecard-
client.de:24728/larms/getAttributes',
      type: 'GET',
      success: function(response) {
        // do success
      },
      error: function(xhr, status, text) {
        // do error
      }
    })
  }
```

Figure 16: Example call to the LARMS plugin with jQuery

The HTTP response is of type JWT in the integration scenario. The JWT, which has the JSON data format, is parsed as follows:

```
var res = response.split('.', 2);
var jwtHeader = $.parseJSON(atob(res[0]));
var jwtClaims= $.parseJSON(atob(res[1]));
```

Figure 17: Parsing of the identity attributes

For the last step we assume, that two HTML input elements exist with identifiers `surname_input` and `given_name_input`. To fill in the form, the following integration code can be used:

```
$('#surname_input').val(jwtClaims['http://www.stork.gov.eu/1.0/surname']);
$('#given_name_input').val(jwtClaims['http://www.stork.gov.eu/1.0/givenName'
]);
```

Figure 18: Fill in form with identity attributes

11.2 Local Secure Exchange Service

The FutureID Client is used as a framework for the Local Secure Exchange Service (LSES). The LSES is used to initiate an ESS session key negotiation using a provided TAN and to decrypt a document that is encrypted with the session key. For this purpose, the OpenNCP portal needs to be adapted in order to support the entering of a TAN, as well as to pass it to the respective FutureID plugin (http://localhost:24727/epsos-ess/setTAN?tan=123456[3]). Furthermore, the decryption task via http://localhost:24727/epsos-ess/decrypt must be implemented in the epSOS Document Retrieval mask. These features have not been implemented on the portal level. However, the standardized search fields for country-specific patient searches might be easily adapted in order to support it.[4]

The underlying FutureID plugin can be invoked via JavaScript using the very same principles that are used for integrating the Signature Service. However, the use of a TAN for authorization and secure exchange means is dependent on the server side platform (NCP-A).

The LSES plugin consist of solely one class `LsesAction` that implements `org.openecard.addon.bind.AppPluginAction`. It forwards requests to the underlying epSOS ESS implementation library [6] that uses the Remote Key Negotiation Service. In this way, the plugin realizes an HTTP binding for the Remote Key Negotiation Service.

- http://localhost:24727/epsos-ess/setTAN?tan=123456 invokes `ESS_B.setTan(123456)`,
- http://localhost:24727/epsos-ess/decrypt returns document via `ESS_B.decryptDocument()` that conforms to RFC 2388 where the MIME-Type is originated from the request.

11.3 Remote Key Negotiation Service

The Remote Key Negotiation Service (RKNS) is invoked by the Local Secure Exchange Service. It relies on the former epSOS ESS library [6] that has been extended in order to use the *Password Authenticated Key Exchange by Juggling* (J-PAKE, just PAKE in subsequent text) protocol [7].

Mainly, the following classes implement this protocol:

- `PakeDriver` drives the message exchange protocol of J-PAKE.
- `PakeParticipant` derives a strong shared session key from a given rather weak TAN.

[3] A secure connection is established via https://www.localhost-ecard-client.de:24728/epsos-ess/setTAN?tan=123456.
[4] Cf. https://openncp.atlassian.net/wiki/display/GUI/PAC%3A+Define+custom+fields+for+user

In order to realize the communication with a PAKE respondent, an endpoint in country A must be set. This is configured in the Java properties file “ess_b_service.properties” with a key/value pair “`ess_a_pake_addr=http://endpoint-country-a:8080/pake`” . Please note that the connection must be secured in productive environments. The OpenNCP circle of trust is technically set up through Virtual Private Network/IPsec. Certificates are distributed via ETSI TS 102 231.

12. Conclusion

The implementation of the Citizen Services Pilot comprises the extension of the OpenNCP portal in order to allow for external authentications as well as eID-based signing. Enabling OpenNCP to use FutureID as an authentication schema was easy, as interfaces rely on well-known and mature standards. The tricky part was to enable OpenNCP to use external authentication schemes in general. This pilot implementation revealed that existing epSOS processes such as the consent-based process could be improved due to an enhanced patient involvement (i. e. eID-based signature).

13. Appendix A – LARMS Plugin Addon.xml File

The following Addon.xml file is used by the LARMS plugin.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AddonSpecification>
    <ID>LARMS</ID>
    <Version>0.1</Version>
    <About/>
    <License/>
    <LocalizedName xml:lang="EN">Local Attribute Retrieval and Mapping
Service</LocalizedName>
    <LocalizedDescription xml:lang="EN">This plugin was written in the
FutureID project context, in order to draw a connection to the epSOS
project. Its task is to map the publicly available personal data of European
healthcare cards to a predefined attribute set.</LocalizedDescription>
    <Logo/>
    <ConfigDescription>
        <Entries>
        </Entries>
    </ConfigDescription>
    <BindingActions>
        <AppPluginSpecification>
<ClassName>org.openecard.plugins.larms.ExtendedPluginAction</ClassName>
            <LocalizedName xml:lang="DE"></LocalizedName>
            <LocalizedDescription xml:lang="DE"></LocalizedDescription>
            <LocalizedName xml:lang="EN"></LocalizedName>
            <LocalizedDescription xml:lang="EN"></LocalizedDescription>
            <!-- That's the subpath to the domain e.g.
http://localhost:27242/larms -->
            <ResourceName>larms/getAttributes</ResourceName>
            <ConfigDescription/>
        </AppPluginSpecification>
    </BindingActions>
    <ApplicationActions/>
    <IFDActions/>
    <SALActions/>
</AddonSpecification>
```

Figure 19: Example Addon.xml file used by the LARMS plugin

14. Appendix B – Sample OASIS DSS Request for German-speaking Patients

```
<dss:SignRequest RequestID="ConsentSignatureRequest"
xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema">
    <dss:OptionalInputs>
        <dss:SignatureType>urn:ietf:rfc:3275</dss:SignatureType>

<dss:SignatureForm>urn:oasis:names:tc:dss:1.0:profiles:AdES:forms:BES</dss:S
ignatureForm>
        <dss:SignaturePlacement WhichDocument="consent">
            <dss:XPathAfter>/*/*[local-name()='component']</dss:XPathAfter>
        </dss:SignaturePlacement>
        <dss:SignedReference WhichDocument="consent">
            <ds:Transforms xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
                <ds:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-
19991116">
                    <xsl:stylesheet id="stylesheet" version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:cda="urn:hl7-org:v3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                        <xsl:output encoding="UTF-8" indent="yes"
method="html" version="1.0" />
                        <xsl:template match="/">
                            <xsl:element name="html"
namespace="http://www.w3.org/1999/xhtml">
                                <xsl:element name="head">
                                    <xsl:element name="meta">
                                        <xsl:attribute name="http-
equiv">content-type</xsl:attribute>
                                        <xsl:attribute
name="content">text/html</xsl:attribute>
                                    </xsl:element>
                                    <xsl:element name="style">
                                        <xsl:attribute
name="type">text/css</xsl:attribute>
                                        <xsl:attribute
name="media">screen</xsl:attribute>
                                        <xsl:text>                    ul {
list-style-type: square                    }
body{                               font-family:sans-serif;
font-size:12pt;                             line-height:normal;
margin:10px                         }
.title{                             font-family:sans-serif;
font-weight:bold;                               text-decoration:underline;
font-size:16pt;                             text-align:center
}                       .section{                               font-
family:sans-serif;                      }                   </xsl:text>
                                    </xsl:element>
                                </xsl:element>
```

```
                                <xsl:element name="body">
                                    <xsl:apply-templates
select="cda:ClinicalDocument" />
                                </xsl:element>
                            </xsl:element>
                        </xsl:template>
                        <xsl:template match="cda:ClinicalDocument">
                            <xsl:element name="div">
                                <xsl:element name="div">
                                    <xsl:attribute
name="class">title</xsl:attribute>
                                    <xsl:text>Patienteneinwilligung für den
Abruf einer epSOS Patientenkurzakte (Patient Summary)</xsl:text>
                                </xsl:element>
                                <xsl:element name="div">
                                    <xsl:element name="p">
                                        <xsl:text>Für</xsl:text>
                                    </xsl:element>
                                    <xsl:element name="ul">
                                        <xsl:element name="li">
                                            <xsl:call-template
name="getNameInFull">
                                                <xsl:with-param name="name"
select="/cda:ClinicalDocument/cda:recordTarget/cda:patientRole/cda:patient/c
da:name"/>
                                                <xsl:with-param
name="gender"
select="/cda:ClinicalDocument/cda:recordTarget/cda:patientRole/cda:patient/c
da:administrativeGenderCode"/>
                                            </xsl:call-template>
                                            <xsl:text> (Pat.-Nr: </xsl:text>
                                            <xsl:value-of
select="/cda:ClinicalDocument/cda:recordTarget/cda:patientRole/cda:id/@exten
sion"/>
                                            <xsl:text>), geboren am
</xsl:text>
                                            <xsl:call-template
name="formatDate">
                                                <xsl:with-param name="date"
select="/cda:ClinicalDocument/cda:recordTarget/cda:patientRole/cda:patient/c
da:birthTime/@value"/>
                                            </xsl:call-template>
                                            <xsl:text>, wohnhaft in
</xsl:text>
                                            <xsl:call-template
name="getContact">
                                                <xsl:with-param
name="contact"
select="/cda:ClinicalDocument/cda:recordTarget/cda:patientRole"/>
                                            </xsl:call-template>
                                        </xsl:element>
                                        <xsl:element name="li">
```

```
                                        <xsl:text>mit dem Einsatzzweck:
Behandlung</xsl:text>
                                    </xsl:element>
                                    <xsl:choose>
                                        <xsl:when
test="//cda:templateId[@root='1.3.6.1.4.1.19376.1.5.3.1.2.6']/../cda:effecti
veTime/cda:high">
                                            <xsl:element name="li">
                                                <xsl:text>bis spätestens
zum </xsl:text>
                                                <xsl:call-template
name="formatDate">
                                                    <xsl:with-param
name="date"
select="//cda:templateId[@root='1.3.6.1.4.1.19376.1.5.3.1.2.6']/../cda:effec
tiveTime/cda:high/@value"/>
                                                </xsl:call-template>
                                            </xsl:element>
                                        </xsl:when>
                                    </xsl:choose>
                                </xsl:element>
                                <xsl:element name="p">
                                    <xsl:text>Die Patientenkurzakte
ermöglicht den mich behandelnden Ärzten und unter ärztlicher Aufsicht deren
berufsmäßigen Gehilfen (z.B. Krankenpflegepersonal, medizinisches
Funktionspersonal) auf die folgenden Daten zuzugreifen und diese zu
Behandlungszwecken zu verarbeiten:</xsl:text>
                                </xsl:element>
                                <xsl:element name="ul">
                                    <xsl:element
name="li">Administrative Informationen (z.B. Name, Geburtsdatum,
Adressen)</xsl:element>
                                    <xsl:element name="li">Eine
Zusammenfassung der wichtigsten klinischen Patientendaten (z.B. Allergien,
aktuelle medizinische Probleme, medizinische Implantate, wesentliche
chirurgische Eingriffe während der letzten sechs Monate)</xsl:element>
                                    <xsl:element name="li">Eine
Zusammenfassung aller aktuellen Medikamentenverschreibungen</xsl:element>
                                    <xsl:element name="li">Informationen
über die Patientenkurzakte selbst (z.B. wann und von wem sie erstellt
wurde).</xsl:element>
                                </xsl:element>
                                <xsl:element name="p">
                                    <xsl:text>Ich bin </xsl:text>
                                    <xsl:choose>
                                        <!-- Opt-out -->
                                        <xsl:when
test="//cda:templateId[@root='1.3.6.1.4.1.19376.1.5.3.1.2.6']/../cda:code[@c
ode='1.3.6.1.4.1.12559.11.10.1.3.2.4.1.2']">
                                            <xsl:element name="strong">
                                                <xsl:text>nicht
</xsl:text>
                                            </xsl:element>
```

```
                                        </xsl:when>
                                    </xsl:choose>
                                    <xsl:text>damit einverstanden, dass
die nachfolgend benannten Ärzte und deren berufsmäßige Gehilfen diese
Patientenkurzakte nutzen, wenn und soweit dies im Rahmen meiner Behandlung
erforderlich ist.</xsl:text>
                                </xsl:element>
                                <xsl:element name="ol">
                                    <xsl:for-each
select="cda:custodian/cda:assignedCustodian/cda:representedCustodianOrganiza
tion">
                                        <xsl:element name="li">
                                            <xsl:call-template
name="getNameInFull">
                                                <xsl:with-param
name="name" select="cda:name"/>
                                            </xsl:call-template>
                                            <xsl:text>, </xsl:text>
                                            <xsl:call-template
name="getContact">
                                                <xsl:with-param
name="contact" select="current()"/>
                                            </xsl:call-template>
                                        </xsl:element>
                                    </xsl:for-each>
                                </xsl:element>
                                <xsl:element name="p">
                                    <xsl:text>Ich wurde darüber
aufgeklärt, dass die Patientenkurzakte eine technische Lösung ist, um
mehreren behandelnden Ärzten bzw. Institutionen Zugriff auf die relevanten
Informationen zu ermöglichen, und mir wurde erläutert, wie sie genutzt wird.
Ein entsprechendes Informationsblatt habe ich erhalten und gelesen; den
Inhalt dieses Informationsblattes betrachte ich als Bestandteil dieser
Einwilligung.</xsl:text>
                                </xsl:element>
                            </xsl:element>
                            <xsl:element name="p">
                                <xsl:variable name="city"
select="cda:custodian/cda:assignedCustodian/cda:representedCustodianOrganiza
tion/cda:addr/cda:city" />
                                <xsl:choose>
                                    <xsl:when test="$city">
                                        <xsl:value-of select="$city"/>
                                    </xsl:when>
                                    <xsl:otherwise>
                                        <xsl:text>NO CITY</xsl:text>
                                    </xsl:otherwise>
                                </xsl:choose>
                                <xsl:text>, am </xsl:text>
                                <xsl:call-template name="formatDate">
                                    <xsl:with-param name="date"
select="cda:effectiveTime/@value" />
                                </xsl:call-template>
```

```
                                </xsl:element>
                            </xsl:element>
                        </xsl:template>
                        <xsl:template name="getNameInFull">
                            <xsl:param name="name"/>
                            <xsl:param name="gender"/>
                            <xsl:choose>
                                <xsl:when test="$name/cda:family">
                                    <xsl:choose>
                                        <xsl:when test="$name/cda:prefix">
                                            <xsl:value-of
select="$name/cda:prefix"/>
                                            <xsl:text> </xsl:text>
                                        </xsl:when>
                                        <xsl:otherwise>
                                            <xsl:choose>
                                                <xsl:when
test="$gender/@code='M'">
                                                    <xsl:text>Herr
</xsl:text>
                                                </xsl:when>
                                                <xsl:when
test="$gender/@code='F'">
                                                    <xsl:text>Frau
</xsl:text>
                                                </xsl:when>
                                                <xsl:otherwise/>
                                            </xsl:choose>
                                        </xsl:otherwise>
                                    </xsl:choose>
                                    <xsl:value-of select="$name/cda:given"/>
                                    <xsl:text> </xsl:text>
                                    <xsl:value-of
select="$name/cda:family"/>
                                    <xsl:if test="$name/cda:suffix">
                                        <xsl:text>, </xsl:text>
                                        <xsl:value-of
select="$name/cda:suffix"/>
                                    </xsl:if>
                                </xsl:when>
                                <xsl:otherwise>
                                    <xsl:value-of select="$name"/>
                                </xsl:otherwise>
                            </xsl:choose>
                        </xsl:template>
                        <xsl:template name="getContact">
                            <xsl:param name="contact"/>
                            <xsl:apply-templates
select="$contact/cda:addr"/>
                        </xsl:template>
                        <xsl:template match="cda:addr">
                            <xsl:for-each select="cda:streetAddressLine">
                                <xsl:value-of select="."/>
```

```
                            <xsl:text>, </xsl:text>
                        </xsl:for-each>
                        <xsl:if test="cda:streetName">
                            <xsl:value-of select="cda:streetName"/>
                            <xsl:text> </xsl:text>
                            <xsl:value-of select="cda:houseNumber"/>
                            <xsl:text>, </xsl:text>
                        </xsl:if>
                        <xsl:value-of select="cda:postalCode"/>
                        <xsl:text> </xsl:text>
                        <xsl:value-of select="cda:city"/>
                        <xsl:if test="cda:state">
                            <xsl:text>, </xsl:text>
                            <xsl:value-of select="cda:state"/>
                        </xsl:if>
                    </xsl:template>
                    <xsl:template name="formatDate">
                        <xsl:param name="date"/>
                        <xsl:choose>
                            <xsl:when test="substring ($date, 7,
1)='0'">
                                <xsl:value-of select="substring ($date,
8, 1)"/>
                            </xsl:when>
                            <xsl:otherwise>
                                <xsl:value-of select="substring ($date,
7, 2)"/>
                            </xsl:otherwise>
                        </xsl:choose>
                        <xsl:text>. </xsl:text>
                        <xsl:variable name="month" select="substring
($date, 5, 2)"/>
                        <xsl:choose>
                            <xsl:when test="$month='01'">
                                <xsl:text>Januar </xsl:text>
                            </xsl:when>
                            <xsl:when test="$month='02'">
                                <xsl:text>Februar </xsl:text>
                            </xsl:when>
                            <xsl:when test="$month='03'">
                                <xsl:text>März </xsl:text>
                            </xsl:when>
                            <xsl:when test="$month='04'">
                                <xsl:text>April </xsl:text>
                            </xsl:when>
                            <xsl:when test="$month='05'">
                                <xsl:text>Mai </xsl:text>
                            </xsl:when>
                            <xsl:when test="$month='06'">
                                <xsl:text>Juni </xsl:text>
                            </xsl:when>
                            <xsl:when test="$month='07'">
                                <xsl:text>Juli </xsl:text>
```

```
                                </xsl:when>
                                <xsl:when test="$month='08'">
                                    <xsl:text>August </xsl:text>
                                </xsl:when>
                                <xsl:when test="$month='09'">
                                    <xsl:text>September </xsl:text>
                                </xsl:when>
                                <xsl:when test="$month='10'">
                                    <xsl:text>Oktober </xsl:text>
                                </xsl:when>
                                <xsl:when test="$month='11'">
                                    <xsl:text>November </xsl:text>
                                </xsl:when>
                                <xsl:when test="$month='12'">
                                    <xsl:text>Dezember </xsl:text>
                                </xsl:when>
                            </xsl:choose>
                            <xsl:value-of select="substring ($date, 1, 4)"/>
                        </xsl:template>
                    </xsl:stylesheet>
                </ds:Transform>
                <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#" />
            </ds:Transforms>
        </dss:SignedReference>
    </dss:OptionalInputs>
    <dss:InputDocuments>
        <dss:Document ID="consent" RefURI="#consent1"
RefType="DSSReference">
            <dss:InlineXML ignorePIs="false">
                <ClinicalDocument xmlns="urn:hl7-org:v3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:hl7-org:v3 CDA.xsd">
                    <typeId root="2.16.840.1.113883.1.3"
extension="POCD_MT000040"/>
                    <templateId root="1.3.6.1.4.1.19376.1.5.3.1.1.1"/>
                    <templateId root="1.3.6.1.4.1.19376.1.5.3.1.1.7"/>
                    <id extension="2.16.17.710.803.1000.990.1.667"
root="2.16.470.1.100.1.1.1000.990.1"/>
                    <code codeSystemName="LOINC"
codeSystem="2.16.840.1.113883.6.1" code="57016-8" displayName="Privacy
Policy Acknowledgement Document"/>
                    <title>Consent to Share Information</title>
                    <effectiveTime value="20151026" />
                    <confidentialityCode code="N"
codeSystem="2.16.840.1.113883.5.25"/>
                    <languageCode code="en-US"/>
                    <recordTarget typeCode="RCT">
                        <patientRole classCode="PAT">
                            <id extension="192419714"
root="2.16.470.1.100.1.1.1000.990.1"/>
                            <addr>
                                <streetAddressLine nullFlavor="NI"/>
```

```
                                <city>PORTO</city>
                                <state nullFlavor="NI"/>
                                <postalCode>4200-32 PORTO</postalCode>
                                <country>Portugal</country>
                            </addr>
                            <telecom nullFlavor="NI" />
                            <patient classCode="PSN"
determinerCode="INSTANCE">
                                <name>
                                    <family>PAOLO CERRUTI</family>
                                    <given nullFlavor="NI"/>
                                </name>
                                <administrativeGenderCode code="M"
codeSystem="2.16.840.1.113883.5.1" />
                                <birthTime value="June 13, 1947" />
                                <languageCommunication>
                                    <languageCode code="null"/>
                                </languageCommunication>
                            </patient>
                        </patientRole>
                    </recordTarget>
                    <author typeCode="AUT">
                        <functionCode code="221" displayName="Medical
doctors" codeSystem="2.16.840.1.113883.2.9.6.2.7"
codeSystemName="epSOSHealthcareProfessionalRoles"/>
                        <time value="20151026"/>
                        <assignedAuthor classCode="ASSIGNED">
                            <id extension="null"
root="2.16.470.1.100.1.1.1000.990.1"/>
                            <addr>
                                <streetAddressLine nullFlavor="NI"/>
                                <city nullFlavor="NI"/>
                                <state nullFlavor="NI"/>
                                <postalCode nullFlavor="NI"/>
                                <country nullFlavor="NI"/>
                            </addr>
                            <telecom nullFlavor="NI" />
                            <assignedPerson classCode="PSN"
determinerCode="INSTANCE">
                                <name>
                                    <family nullFlavor="NI"/>
                                    <given nullFlavor="NI"/>
                                </name>
                            </assignedPerson>
                        </assignedAuthor>
                    </author>
                    <custodian typeCode="CST">
                        <assignedCustodian classCode="ASSIGNED">
                            <representedCustodianOrganization
classCode="ORG" determinerCode="INSTANCE">
                                <id root="2.16.470.1.100.1.1.1000.990.1"/>
                                <name>Fraunhofer FOKUS CC eHealth</name>
                                <telecom nullFlavor="NI"/>
```

```
                                <addr>
                                    <country>DE</country>
                                </addr>
                            </representedCustodianOrganization>
                        </assignedCustodian>
                    </custodian>
                    <legalAuthenticator contextControlCode="OP"
typeCode="LA">
                        <time value="20120927112208"/>
                        <signatureCode code="S"/>
                        <assignedEntity classCode="ASSIGNED">
                            <id root="2.16.470.1.100.1.1.1000.990.1"/>
                            <telecom nullFlavor="NI"/>
                            <assignedPerson>
                                <name>
                                    <family>Max</family>
                                    <given>Mustermann</given>
                                </name>
                            </assignedPerson>
                            <representedOrganization classCode="ORG"
determinerCode="INSTANCE">
                                <id root="2.16.470.1.100.1.1.1000.990.1"/>
                                <name>Sample Hospital</name>
                                <addr use="PST">

<streetAddressLine>N/A</streetAddressLine>
                                    <city>CC eHealth Fraunhofer FOKUS</city>
                                    <postalCode>N/A</postalCode>
                                    <state nullFlavor="UNK"/>
                                    <country>DE</country>
                                </addr>
                            </representedOrganization>
                        </assignedEntity>
                    </legalAuthenticator>
                    <documentationOf typeCode="DOC">
                        <serviceEvent classCode="ACT" moodCode="EVN">
                            <templateId
root="1.3.6.1.4.1.19376.1.5.3.1.2.6"/>
                            <id root="2.16.17.710.803.1000.990.1.668"/>
                            <code code="1.3.6.1.4.1.12559.11.10.1.3.2.4.1.1"
codeSystem="1.3.6.1.4.1.12559.11.10.1.3.2.4.1" codeSystemName="Connect-a-
thon eventCodeList" displayName="Opt-in"/>
                            <effectiveTime>
                                <low value="20151026"/>
                                <high value="20151027"/>
                            </effectiveTime>
                        </serviceEvent>
                    </documentationOf>
                    <component>
                        <structuredBody>
                            <component>
                                <section>
                                    <title>BPPC Description</title>
```

```
                                        <text>
                                            <br/>
                                            <content>Policy:
1.3.6.1.4.1.12559.11.10.1.3.2.4.1.1,Description: Opt-in,EffectiveTime From:
20151026, EffectiveTime To: 20151027</content>
                                        </text>
                                    </section>
                                </component>
                            </structuredBody>
                        </component>
                    </ClinicalDocument>
                </dss:InlineXML>
            </dss:Document>
        </dss:InputDocuments>
</dss:SignRequest>
```

Figure 20: Sample OASIS DSS request sent to the eSign plugin

15. Appendix C – Integration of FutureID Authentication with OpenNCP

```
package de.fraunhofer.fokus.ehealth.openncp;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.liferay.portal.NoSuchUserException;
import com.liferay.portal.kernel.exception.PortalException;
import com.liferay.portal.kernel.exception.SystemException;
import com.liferay.portal.kernel.log.Log;
import com.liferay.portal.kernel.log.LogFactoryUtil;
import com.liferay.portal.kernel.servlet.SessionErrors;
import com.liferay.portal.kernel.util.ParamUtil;
import com.liferay.portal.kernel.util.Validator;
import com.liferay.portal.model.User;
import com.liferay.portal.security.auth.AutoLogin;
import com.liferay.portal.security.auth.AutoLoginException;
import com.liferay.portal.service.UserLocalServiceUtil;
import com.liferay.portal.util.PortalUtil;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;
import sun.misc.BASE64Decoder;

public class OpenNcpLoginFilter implements AutoLogin {

    private static Log _log = LogFactoryUtil.getLog(OpenNcpLoginFilter.class);

    public String[] handleException(HttpServletRequest paramHttpServletRequest,
            HttpServletResponse paramHttpServletResponse,
            Exception paramException) throws AutoLoginException {

        _log.info("OpenNcpLoginFilter\n" + paramException);

        return null;
    }

    // Demo Patient:
    //  max.mustermann@openncp.org with password test123. (userId: 15201, companyId: 10254)
    public String[] login(HttpServletRequest paramHttpServletRequest,
            HttpServletResponse paramHttpServletResponse)
            throws AutoLoginException {

        String[] credentials = doLogin(paramHttpServletRequest, paramHttpServletResponse);
```

```
        // redirect to the following page (relative path e.g. /group/myorganization ) in
case of successfull credentials
        if (!(Validator.isNull(credentials) || credentials[0] == null)) {
            paramHttpServletRequest.setAttribute(AutoLogin.AUTO_LOGIN_REDIRECT_AND_CONTINUE,
PortalUtil.getPathMain());
        }

        return credentials;
    }

    private String[] doLogin(HttpServletRequest req, HttpServletResponse res) {

        String samlResponseB64 = ParamUtil.getString(req, "SAMLResponse");

        if (samlResponseB64.length() < 1) {
            return null;
        }

        final BASE64Decoder decoder = new BASE64Decoder();
        final String samlResponse;
        try {
            samlResponse = new String(decoder.decodeBuffer(samlResponseB64), "UTF-8");
        } catch (IOException ex) {
            _log.error("Failed to decode Base64-encoded SAMLReponse.");
            return null;
        }

        final Document samlDoc = parseSamlResponse(samlResponse);

        if (isErrorResponse(samlDoc)) {
            final String errorMessage = getErrorMessage(samlDoc);
            SessionErrors.add(req, "eIDError");
            req.getSession().setAttribute("eIDError", errorMessage);
            return null;
        }

        final String userId = extractUserId(samlDoc);

        _log.debug("Received authenticated eIdentifier :" + userId);

        User user = checkWhetherUserExists(String.valueOf(userId), req);

        if (user != null) {
            _log.debug("Found user id " + user.getUserId() + " - " + user.getEmailAddress()
+ " - " + user.getScreenName());
            return new String[]{String.valueOf(user.getUserId()),
                user.getPassword(),
                String.valueOf(user.isPasswordEncrypted())};
        } else {
            _log.info("eID Authentication succeeded. However, the eIdentifier is not
associated with a User Profile.");
            SessionErrors.add(req, "eIDError");
            req.getSession().setAttribute("eIDError", "eID Authentication succeeded.
However, the eIdentifier is not associated with a User Profile.");
        }
```

```
        return null;
    }

    private static Document parseSamlResponse(final String samlResponse) {
        final DocumentBuilderFactory domFactory = DocumentBuilderFactory.newInstance();
        domFactory.setNamespaceAware(true);

        try {
            final DocumentBuilder db = domFactory.newDocumentBuilder();
            return db.parse(new ByteArrayInputStream(samlResponse.getBytes("UTF-8")));

        } catch (final ParserConfigurationException pce) {
            throw new RuntimeException("Failed to create instance of
org.w3c.dom.DocumentBuilder when parsing SAML Response.", pce);
        } catch (final SAXException se) {
            throw new RuntimeException("Failed to parse SAML Response.", se);
        } catch (final IOException se) {
            throw new RuntimeException("Failed to parse SAML Response.", se);
        }
    }

    /**
     *
     * @param samlDocument
     * @return null, if extraction fails.
     */
    private static String extractUserId(final Document samlDocument) {
        final XPath xpath = XPathFactory.newInstance().newXPath();
        try {
            return (String) xpath.evaluate("//*[local-name()='Attribute' and
@Name='http://www.futureid.org/att/eIdentifier']/*[local-name()='AttributeValue']/text()",
samlDocument, XPathConstants.STRING);
        } catch (final XPathExpressionException ex) {
            _log.error("Failed to extract eIdentifier from SAML Response message.");
            return null;
        }
    }

    private static boolean isErrorResponse(final Document samlDocument) {
        final XPath xpath = XPathFactory.newInstance().newXPath();
        try {
            String status = (String) xpath.evaluate("//*[local-name()='StatusCode']/@Value",
samlDocument, XPathConstants.STRING);
            return !"urn:oasis:names:tc:SAML:2.0:status:Success".equals(status);
        } catch (final XPathExpressionException ex) {
            _log.error("Failed to extract Status from SAML Response message.");
            return true;
        }
    }

    private static String getErrorMessage(final Document samlDocument) {
        final XPath xpath = XPathFactory.newInstance().newXPath();
        try {
            return (String) xpath.evaluate("//*[local-name()='StatusMessage']/text()",
samlDocument, XPathConstants.STRING);
        } catch (final XPathExpressionException ex) {
            _log.error("Failed to extract StatusMessage from SAML Response message.");
```

```
            return null;
        }
    }

    private User checkWhetherUserExists(String authenticatedUserId, HttpServletRequest req)
{

        if (authenticatedUserId == null || authenticatedUserId.isEmpty()) {
            return null;
        }

        long companyId = 0;
        try {

            companyId = PortalUtil.getCompany(req).getCompanyId();

            User user = UserLocalServiceUtil.getUserByOpenId(companyId,
authenticatedUserId);
            if (user != null) {
                return user;
            }

        } catch (NoSuchUserException e) {
            _log.info("No user exists with the key {companyId="
                    + companyId + ", authenticatedUserId=" + authenticatedUserId
                    + "}");
        } catch (PortalException e) {
            _log.info("Could not check existence of user with the key {companyId="
                    + companyId + ", authenticatedUserId=" + authenticatedUserId
                    + "}");
        } catch (SystemException e) {
            _log.info("Could not check existence of user with the key {companyId="
                    + companyId + ", authenticatedUserId=" + authenticatedUserId
                    + "}");
        }
        return null;
    }

}
```

Figure 21: Java implementation of Login Filter.