Department Service Integration with e-Pramaan

Integration document - JAVA specific

Centre for Development of Advanced Computing

Gulmohar Cross Road No. 9, Juhu, Mumbai 400 049.
Telephone: +91 22 2620 1606, +91 22 2620 1574,
Fax: +91 22 2621 0139, +91 22 2623 2195
Website: www.cdac.in
Content
Revision History.................................................................................................................. 2
Abbreviations ..................................................................................................................... 2
1. Introduction .................................................................................................................... 3
2. Process Flow .................................................................................................................. 3
   1. Single Sign On ............................................................................................................. 3
      A. Login (SSO) Initiated at Department Service ......................................................... 3
      B. Login (SSO) Initiated at e-Pramaan ........................................................................ 4
   2. Single Logout (SLO) .................................................................................................. 5
      A. Single LogOut (SLO) Initiated by Department service ......................................... 5
      B. Single LogOut (SLO) Initiated by e-Pramaan ......................................................... 5
3. Required steps for integration ....................................................................................... 6
   A. Add the integration jars to your project .................................................................. 7
   B. Copy the configuration file epramaanconnector.properties to folder WEB-INF/classes ................................................................. 7
   C. POSTing requests to e-Pramaan .............................................................................. 7
   D. Modify authentication –“Login using e-Pramaan” ................................................. 7
   E. Processing SAML Response .................................................................................... 8
   F. Supporting Single LogOut (SLO) ............................................................................. 10
   G. Processing SLO Response ....................................................................................... 11
   H. Webservice Handlers ............................................................................................... 12
      i. SLO request handler ............................................................................................... 12
      ii. SSO failhandler .................................................................................................... 13
      iii. Logoutfailure handler ......................................................................................... 15
   I. Processing OneTimeVerification ............................................................................. 16
   J. Session Management .............................................................................................. 17
   K. Web.xml .................................................................................................................. 17
4. Modify epramaan.properties ....................................................................................... 17
5. Register Department Service at e-Pramaan Service Portal ....................................... 18
6. Problems Faced During Integration ........................................................................ 20
©Centre for Development of Advanced Computing
Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Reason for Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 (Draft)</td>
<td>22-05-15</td>
<td>e-Pramaan Integration Team</td>
<td>e-Pramaan Senior Project Manager</td>
</tr>
<tr>
<td>1.3 (Draft)</td>
<td>12-06-15</td>
<td>e-Pramaan Integration Team</td>
<td>e-Pramaan Senior Project Manager</td>
</tr>
<tr>
<td>1.3.1</td>
<td>09-11-2015</td>
<td>e-Pramaan Integration Team</td>
<td>Added section on common problems faced during integration</td>
</tr>
</tbody>
</table>

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAML</td>
<td>Security Assertion Mark-up Language</td>
</tr>
<tr>
<td>SP</td>
<td>Service Provider (Department)</td>
</tr>
<tr>
<td>SSO</td>
<td>Single Sign On</td>
</tr>
<tr>
<td>SLO</td>
<td>Single LogOut</td>
</tr>
</tbody>
</table>

Intended Audience

The recommended audience for this document is the enterprising person (system administrator/software developer) responsible for e-Pramaan integration at Department end. This document may be useful for the project manager/Department head to assess the effort required for integration with e-Pramaan.
Prerequisite
The integrating person at Department end should be well versed in web application development using JAVA and familiar with the work flow of the Department Service.

1. Introduction
This document explains the steps involved in integrating Department services developed as JAVA web application with e-Pramaan. It explains the process flow and the step-by-step guidance for integrating JAVA web application with e-Pramaan to achieve Single Sign-On (SSO), Single Logout (SLO) functionality along with the authentication services provided by e-Pramaan.

2. Process Flow

1. Single Sign On
Single Sign-On (SSO) is an access control mechanism across multiple independent software systems. This allows user to log in once and gain access to all related services, without being prompted for log in again at each of them. e-Pramaan allows the user to initiate SSO either from Department Service or from e-Pramaan portal. Similarly, Single Logout session can also be initiated at Department Service or at e-Pramaan portal end.

SAML 2.0 is used for SSO implementation.

A. Login (SSO) Initiated at Department Service
Steps involved in SSO initiated by Department Service are depicted in Figure 2.1.

1. User at Department Service initiates SSO by clicking the option to "Login Using e-Pramaan".
2. Department Service then creates SAML SSO request and forwards the user to e-Pramaan for authentication.
3. User is authenticated by e-Pramaan using Challenge-Response mechanism.
4. User is authenticated successfully on e-Pramaan.
5. The user is redirected back to the initiating Department Service. Since the user has been authenticated at e-Pramaan, the Department Service accepts the user and allows him/her to login.
6. If the user fails to authenticate himself/herself on e-Pramaan, the SAML response returns authentication failure.
B. Login (SSO) Initiated at e-Pramaan

Steps involved in SSO initiated by user at e-Pramaan portal depicted in Figure 2.2.

1. The user directly comes to e-Pramaan portal and logs in using Login and Password.
2. When the user selects a service, e-Pramaan checks the authentication required by the service.
3. According to the requirement of the service, e-Pramaan uses the challenge-response to complete the authentication.
4. The user completes the authentication by providing the appropriate response.
5. e-Pramaan then initiates an SSO session between e-Pramaan and the selected service.
6. The user is then redirected to the selected Department Service where (s)he will be allowed to log in without entering his/her credentials again.
2. Single Logout (SLO)
During every user session at e-Pramaan, user may log-in into multiple services. When the user is logged out from one service, the user is logged out from all active services. This is achieved by initiating Single LogOut (SLO) either from Department Service or e-Pramaan portal. SLO is triggered when the user selects the option to logout, either from the Department Service or from e-Pramaan.

A. Single LogOut (SLO) Initiated by Department service
SLO is initiated by the Department Service when the user decides to logout at the Department Service. The service will Log Out the user locally (or the service may logout the user locally after receiving the response from e-Pramaan) and then creates SAML SLO request to e-Pramaan to initiate SLO broadcast. This will ensure that the user will be logged out from every service the user was logged in through SSO session. This is depicted in Figure 2.3. Note that the service has an option not to accept the SLO token.

![Figure 2.3: SLO Initiated from Department Service](image)

B. Single LogOut (SLO) Initiated by e-Pramaan
Single LogOut (SLO) will be triggered from e-Pramaan when the user selects the option to logout from e-Pramaan portal. e-Pramaan portal will then broadcast the LogOut Request to all the services which are

©Centre for Development of Advanced Computing
logged in through the current active session. On receiving the Single LogOut Request (SLO request) may terminate the local session and notify e-Pramaan the Logout status via logout response message. The scenario is depicted in Figure 2.4.

3. Required steps for integration
The steps involved in integrating a JAVA application with e-Pramaan are listed below. Sample code for this is provided in the integration kit.

- Add the integration jars & dependent jars to your project.
- Copy the configuration file epramaanconnector.properties to WEB-INF/classes.
- Provide link “Login using e-Pramaan” on the login page of Department Service.
- Modify OnClick event of “Login using e-Pramaan” link to authenticate using e-Pramaan.
- Modify/implement logic to consume SSO Token sent by e-Pramaan.
- Modify the logout procedure to direct through e-Pramaan.
- Modify/implement logic to consume logout response from e-Pramaan.
- Implement a RESTful WebService using the template provided in the Integration Kit, to receive SLO broadcast request from e-Pramaan.
- Implement logic for OneTimeVerification as detailed further.
A. Add the integration jars to your project

e-Pramaan integration can be done using the jars provided in the integration kit. Communication between e-Pramaan and the Department Service is managed by `epramaanconnector.jar` and `epramaan-client.jar`. These jars along with dependent jars are provided in the Integration Kit. These must be added to the Department Service application to facilitate integration with e-Pramaan.

<table>
<thead>
<tr>
<th>Serial No</th>
<th>Component</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>epramaanconnector.jar</td>
<td>This component manages communication between Department Service &amp; e-Pramaan</td>
</tr>
<tr>
<td>2</td>
<td>epramaan-client.jar</td>
<td>This component manages de-serialization of SSO Token, sent by e-Pramaan on successful authentication</td>
</tr>
<tr>
<td>3</td>
<td>opensaml-x.x.x.jar</td>
<td>This is the SAML library for creating SAML Requests/Response</td>
</tr>
</tbody>
</table>

B. Copy the configuration file `epramaanconnector.properties` to folder WEB-INF/classes

The configuration file for e-Pramaan integration is the file `epramaanconnector.properties`. This configuration file has to be copied to the folder `WEB-INF/classes` of the Department Service application. It is used by the `epramaanconnector.jar` for getting configuration info regarding e-Pramaan and Department Service like e-Pramaan URL, Department Service ID, encryption seed etc.

C. POSTing requests to e-Pramaan

e-Pramaan can receive authentication request either as GET or POST method, but it prefers POST redirect binding because GET can transport maximum of 8K bytes data. Department Service can implement POST redirect by using `forward.jsp` provided in the Integration Kit. This page receives SAML Request as a parameter from the Department Service and forwards it to e-Pramaan server. This file provided in the integration kit must be copied to folder `WebContent/jsp` of the Department Service during integration.

D. Modify authentication – “Login using e-Pramaan”

- The authentication mechanism in the service application has to incorporate a link to “Login using e-Pramaan”.
- In your application, add a button/link named ‘Login using e-Pramaan’.
- Using the given integration jars, create an authentication request for an event generated by `onclick()` and send it to e-Pramaan for authenticating the user.
Sample code for authentication is given below. Refer SSOSamlRequestCreator.java in the Integration Kit for complete source code.

Table 3.2 : Code Sample for Authentication

```java
//get the current application context
ServletContext context = request.getServletContext();

//Load the e-Pramaan properties file
Properties prop = (Properties) context.getAttribute("EPRAMAAN_PROPERTIES");

//Instantiate SAML connector
SPSamlUtil spSamlUtil = new SPSamlUtilImpl();

//Create the request id. Department service may want to log this in database for future reference
String id = UUID.randomUUID().toString();

String samlReq = null;
//Create the SAML authentication request(signed and encrypted)
try {
    samlReq = spSamlUtil.createSSOSamlRequest(prop, spSamlUtil.getCredentialForSigning(prop), id);
} catch (SamlException e) {
    e.printStackTrace();
}

//Set the SAML authentication request as the request parameter. This will be sent to e-Pramaan for authentication
request.setAttribute("SAMLRequest", samlReq);
RequestDispatcher rd = request.getRequestDispatcher("jsp/forward.jsp");

//Post the login request via jsp file
rd.forward(request, response);
```

E. Processing SAML Response

a. After successful authentication at e-Pramaan the user is redirected to the Department service for which SAML request was initiated. This will be a HTTP POST REDIRECT and the user credentials will be provided by e-Pramaan in the SAML Response. The service has to consume the SAML Response and decide the further logic for allowing the user to access desired service.

b. The SAML Response sent by e-Pramaan will be received by a Servlet at the Department service end. It will decode and process the SAML Response to allow the user login on the service. Sample code for decoding/processing of the SAML Response is given in Table 3.3. Refer consume.java in the integration kit for complete source code.

Table 3.3 : Code sample for SSO response processing

```java
//get the current application context
ServletContext context = request.getServletContext();
```
```java
//initialize sp-epramaan connector component
SPSamlUtil spSamlUtil = new SPSamlUtilImpl();

//Get the current session
HttpSession session = request.getSession();

//Get the custom session manager
SSOSessionManager sessionManager = (SSOSessionManager)
    context.getAttribute("SSO_MANAGE");

//load the epramaan properties file
Properties prop = (Properties)
    context.getAttribute("EPRAMAAN_PROPERTIES");

//Get the response string, received from e-Pramaan
String samlResponse = request.getParameter("SAMLResponse");
SpSSOResponse idpSSOResponse = null;
try {
    //Get the SSOResponse object from the response received
    idpSSOResponse = spSamlUtil.getSSOSamlResponseObject(prop, samlResponse);

    //Get session index from response
    String sessionIndex = idpSSOResponse.getSessionIndex();

    //save the session index
    sessionManager.addSession(sessionIndex, session);

    //Get SSOToken from response object
    SSOToken ssoToken = idpSSOResponse.getSsoToken();
    if (ssoToken.getUserSPServiceDetail() != null &&
        ssoToken.getUserSPServiceDetail().getServiceUserId() != null) {
        //Set sp username received in token to session
        session.setAttribute("epramaanId", ssoToken.getUserSPServiceDetail().getServiceUserId());
    } else {
        //Set Adhaar number received in token.
        session.setAttribute("epramaanId", ssoToken.getAadhaarNumber());
    }
}
catch (SamlException e) {
    // -- Fail case manage
    page = "/jsp/Fail.jsp";
    catch (SamlException e) {
        // -- Fail case manage
        page = "/jsp/Fail.jsp";
        e.printStackTrace();
    }
}

//--Forward to some page on success and failure
RequestDispatcher dispatcher =
    getServletContext().getRequestDispatcher(page);
dispatcher.forward(request, response);
```
**F. Supporting Single LogOut (SLO)**

A user logged in through e-Pramaan will be able to log-out using *Single LogOut (SLO)* feature, which implies *SLO request* initiated will Log Out user from all the active services for that user.

Whenever a user initiates a logout at service, a *SAML SLO request* will be sent to e-Pramaan by the Service. The service will have to provide a "Logout" button to incorporate this feature. Sample code for creating *SAML SLO request* is given in Table 3.4. Refer *SLOSamlLogoutCreator.java* in the integration kit for sample source code.

**Table 3.4 : Code sample for creating SLO request**

```java
//Get the current session
HttpSession session = request.getSession();

//get application Context
ServletContext context = request.getServletContext();

//load the properties file
Properties prop = (Properties) context.getAttribute("EPRAMAAN_PROPERTIES");

//set Service ID as nameID
String nameId = (String) session.getAttribute("epramaanId");

//Remove the session (as soon as the user clicked the logout button, he/she
//should be logged out)
SSOSessionManagersessionManager = (SSOSessionManager)
getServletContext().getAttribute("SSO_MANAGE");

//Get the Session Index of the current session
String sessionIndex = sessionManager.getKeyByValue(session);

//Remove the current session
sessionManager.removeSession(sessionIndex);

//Create logout request and send back to e-Pramaan. This will initiate SLO at
//e-Pramaan
String slosamlRequest = null;
try {
    slosamlRequest = spSamlUtil.createSLOSamlRequest(prop,
spSamlUtil.getCredentialForSigning(prop), nameId, sessionIndex);
} catch (SamlException e) {
    e.printStackTrace();
}

//URL encode the request if send by GET
String URLEncoded = URLEncoder.encode(slosamlRequest,"UTF-8");

//Create the Logout URL
String ePramaanURL =
prop.getProperty("ePramaanURL") + prop.getProperty("SingleLogoutServiceURL") +
"?SAMLRequest=" + URLEncoded;
ePramaanURL=" + ServiceId=" + prop.getProperty("Issuer");
```
G. Processing SLO Response

SLO Response will be sent by e-Pramaan to the initiating Department service in response to its SLO Request. It contains the status whether SLO was a success/failure at e-Pramaan. This SLO Response will be sent via "POST Redirect" to the initiating Department Service.

- To consume the SLO Response, the Service has to implement a listener to receive and process the SLO Response.
- The SLO Response will have a status success if the logout was successful & status as Request failed if the logout was not successful.
- This listener will receive SLO Response, process it & alert the user whether the LogOut operation was successful at e-Pramaan or not.

Sample code for creating SLO Response is given in Table 3.5. Refer SLOLogoutResponseConsumer.java for complete source code.

Table 3.5 : Code sample for SLO Response

```java
//Get current context
ServletContext context = request.getServletContext();

//Initialize the SAML Connector
SPSamlUtilsamlUtil = new SPSamlUtilImpl();

//Load the properties file
Properties prop = (Properties) context.getAttribute("EPRAMAAN_PROPERTIES");

//Get the SLO Response(SLO Response) from e-Pramaan
String samlResponse = request.getParameter("SAMLResponse");
SpSLOResponse idpSLOResponse = null;
//Get the logout response object
try {
    idpSLOResponse = samlUtil.getSLOSamlResponseObject(prop, samlResponse);
    //Check the logout status returned by e-Pramaan
    if (idpSLOResponse.getStatus().compareTo(org.opensaml.saml2.core.StatusCode.SUCCESS_URI) == 0) {
        path = "/jsp/LogoutSuccessfully.jsp";
    } else {
        path = "/jsp/LogoutFail.jsp";
    }
} catch (SamlException e) {
    path = "/jsp/LogoutFail.jsp";
    e.printStackTrace();
}

//Redirect the user to respective page
```
H. Webservice Handlers

i. SLO request handler

e-Pramaan allows user to log in to multiple services through e-Pramaan, in a single user session at e-Pramaan. Suppose that the user is logging out at the e-Pramaan portal, the user should logout from all the associated services for Single LogOut. This is called e-Pramaan initiated Single Logout Service (SLO). e-Pramaan initiated SLO is implemented only through RESTful web services.

In e-Pramaan initiated SLO, Department Service will receive a Logout Request from ePramaan. The service has to process and validate the request. On successful processing of the request, the Department Service logs out the user by terminating the user session. After this, the Service has to send the status of the logout in a synchronous REST service response.

The integrating Service has to implement the RESTful web service for receiving the SLO request from e-Pramaan. Sample code for Service "SLO Request consumer" is given in Table 3.6.

Table 3.6 : Code sample for receiving SLO Broadcast

```java
//initialize SAML Connector
SPSamlUtil spSamlUtil = new SPSamlUtilImpl();

//Get the current context
ServletContext context = request.getServletContext();

String logoutResponseXML = null;

//Load the properties file
Properties prop = (Properties) context.getAttribute("EPRAMAAN_PROPERTIES");

InputStream stream1 = null;
try {
    //Receive the logout request sent by e-Pramaan on WebService
    stream1 = request.getInputStream();
} catch (IOException e1) {
    e1.printStackTrace();
}

BufferedReader bRead = new BufferedReader(new InputStreamReader(stream1));
String temp = null;
String SAMLRequest = "";
try {
    while ((temp = bRead.readLine()) != null) {
        SAMLRequest = SAMLRequest + temp;
    }
} catch (IOException e1) {
    e1.printStackTrace();
}
```
```java
try {
    idpSLORequest = spSamlUtil.getSLOSamlRequestObject(prop, SAMLRequest);
}

//Get the sessionIndex from SLO Request
String sessionIndex = idpSLORequest.getSessionIndex();

//Delete the local session only if the request was validated and no error occurred in between
SSOSessionManagersessionManager = (SSOSessionManager) context.getAttribute("SSO_MANAGE");
sessionManager.removeSession(sessionIndex);

//Create SLO Response XML to send to e-Pramaan
try {
    logoutResponseXML = spSamlUtil.createSLOSamlResponse(prop,
               spSamlUtil.getCredentialForSigning(prop), idpSLORequest.getRequestId(),
               errorOccured);
}

//get the response output stream
ServletOutputStream out = response.getOutputStream();
try {
    //write the response to the response output stream
    out.write(logoutResponseXML.getBytes());
    out.flush();
    out.close();
}
```
sample for this is given in Table 3.7. Refer `SSOAuthFailResponseConsumer.java` in the integration kit for the complete source code.

```
//Get the context
ServletContext context = request.getServletContext();

//Initiale SAML Connector
SPSamlUtil spSamlUtil = new SPSamlUtilImpl();
InputStream stream1 = null;

//Load the properties file
Properties prop = (Properties) context.getAttribute("EPRAMAAN_PROPERTIES");
try {
    //Receive the Response from e-Pramaan, sent on WebService
    stream1 = request.getInputStream();
} catch (IOException e1) {
    e1.printStackTrace();
}

//Get the reader for the stream
BufferedReader bRead = new BufferedReader(new InputStreamReader(stream1));
String SAMLResponse = "";
String temp = null;

//Read the stream to get SAML Response
try {
    while ((temp = bRead.readLine()) != null) {
        SAMLResponse = SAMLResponse + temp;
    }
} catch (IOException e1) {
    e1.printStackTrace();
}

//Get the Response Object
SpSSOResponse idpSSOResponse = null;
try {
    idpSSOResponse = spSamlUtil.getSSOSamlResponseObject(prop, SAMLResponse);
} catch (SamlException e) {
    e.printStackTrace();
}

//Get the Status - reason for SSO Fail; Log it in DB
System.out.println(idpSSOResponse.getStatus());
```
### iii. Logout failure handler

When the user clicks the LogOut button at the Service, it redirects the user to e-Pramaan due to implemented SLO facility. When LogOut attempt is successful, the user is redirected back to the initiating Service. But, when LogOut fails at e-Pramaan, the user is not redirected back. The LogOut failure, in this case, is intimated via RESTful web service logoutfail Web Service.

The Service to implement the RESTful web service handlers as mentioned above. Sample code is given in Table 3.8. Refer `SLOLogoutFailResponseConsumer.java` in Integration Kit for the complete source code.

<table>
<thead>
<tr>
<th>Table 3.8: Code sample for SLO failure handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>//Get the context</td>
</tr>
<tr>
<td>ServletContext context = request.getServletContext();</td>
</tr>
<tr>
<td>//Load the properties file</td>
</tr>
<tr>
<td>Properties prop = (Properties) context.getAttribute(&quot;EPRAMAAN_PROPERTIES&quot;);</td>
</tr>
<tr>
<td>InputStream stream1 = null;</td>
</tr>
<tr>
<td>//Initialize SAML Connector</td>
</tr>
<tr>
<td>SPSamlUtil spSamlUtil = new SPSamlUtilImpl();</td>
</tr>
<tr>
<td>try {</td>
</tr>
<tr>
<td>//Receive the SLO Response from e-Pramaan, sent on WebService</td>
</tr>
<tr>
<td>stream1 = request.getInputStream();</td>
</tr>
<tr>
<td>} catch (IOException e1) {</td>
</tr>
<tr>
<td>e1.printStackTrace();</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>//Get the reader for the stream</td>
</tr>
<tr>
<td>BufferedReader bRead = new BufferedReader(new InputStreamReader(stream1));</td>
</tr>
<tr>
<td>String SAMLResponse = &quot;&quot;;</td>
</tr>
<tr>
<td>String temp = null;</td>
</tr>
<tr>
<td>//Read the stream to get SLO Response</td>
</tr>
<tr>
<td>try {</td>
</tr>
<tr>
<td>while ((temp = bRead.readLine()) != null) {</td>
</tr>
<tr>
<td>SAMLResponse = SAMLResponse + temp;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>} catch (IOException e1) {</td>
</tr>
<tr>
<td>e1.printStackTrace();</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>System.out.println(&quot;SLO fail response received = &quot; + SAMLResponse);</td>
</tr>
<tr>
<td>//Get the SLO Response Object</td>
</tr>
<tr>
<td>SpSLOResponse idpSLOResponse = null;</td>
</tr>
<tr>
<td>try {</td>
</tr>
<tr>
<td>idpSLOResponse = spSamlUtil.getSLOSamlResponseObject(prop, SAMLResponse);</td>
</tr>
<tr>
<td>} catch (SamlException e) {</td>
</tr>
<tr>
<td>e.printStackTrace();</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>//Get the Status - reason for SLO Fail; Log it in DB</td>
</tr>
<tr>
<td>System.out.println(idpSLOResponse.getStatus());</td>
</tr>
</tbody>
</table>
I. Processing OneTimeVerification

User mapping between e-Pramaan and Department Service can be done by either of the two methods explained below.

1. Adhaar Number based – when the user is uniquely identified at the Department Service using Adhaar Number, this mapping will be used. On successful authentication at e-Pramaan, the user will be redirected to the Department Service along with his/her Adhaar Number. Service then checks whether an account for this Adhaar number exists. If found, user is allowed to login & access the service. Otherwise user will have to register at the Department Service before he is allowed to login using e-Pramaan.

2. Service UserID Seeding at e-Pramaan - Service UserID is the UserID used at the Department Service, using which the user logs into the Department Service.

If at the service end, Aadhaar seeding is not done for all the users of the service, the service will go for the option Department UserID seeding at e-Pramaan. If the user is mapped at e-Pramaan using Service User-ID Seeding at e-Pramaan, then at the first login attempt the user will be prompted to do the OneTimeVerification. On successful completion of OneTimeVerification existing Service User-ID will be linked to his e-Pramaan account.

It is mandatory for the service to provide OneTimeVerificationURL at the time of adding a service on portal of e-Pramaan. The logic of OneTimeVerification will validate the user for the first time at the Service end, and push the verification status along with the ServiceUserID to e-Pramaan RESTWebService and complete the ServiceUserId – e-Pramaan account mapping.

When a user tries to access a Service for the first time then e-Pramaan HTTP POST redirects the user to Department service’s OneTimeVerificationURL with three request parameters. Request parameters are defined in Table 3.9 below:

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Parameter name</th>
<th>Parameter value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ssoToken</td>
<td>AES encrypted JSON object of ssotoken</td>
</tr>
<tr>
<td>2</td>
<td>transactionId</td>
<td>Id to uniquely identify the transaction</td>
</tr>
<tr>
<td>3</td>
<td>source</td>
<td>Constant Value i.e.”ePramaan”</td>
</tr>
</tbody>
</table>

Once redirected to the service, the service will ask the user to enter his login credentials. The Service will verify these credentials and the status is pushed back to ePramaan via WebService.

To push user’s enrolment response the e-Pramaan Web service URL is:

https://up.epramaan.in/rest/epramaan/enrol/response. After this, user can access the SP Service by authenticating at e-Pramaan.

©Centre for Development of Advanced Computing
J. Session Management
We have created a class for session management, which is given in the Integration Kit (SSOSessionManager.java). Kindly refer SSOSamlResponseConsumer.java for code sample on session handling.

K. Web.xml
File Web.xml at Department Service must be updated to include the Servlet & WebService Mapping. Sample Web.xml is present in the Integration Kit which the Service can refer to create the required listener mapping.

4. Modify epramaan.properties

epramaan.properties is the key configuration file used by the SAML Connector for integrating with e-Pramaan. The entries in File epramaan.properties are explained as given in Table 3.10 below.

<table>
<thead>
<tr>
<th>Sr. #</th>
<th>Parameter in SSO.properties</th>
<th>Expected Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1*</td>
<td>ePramaanURL</td>
<td><a href="https://up.eparmaan.in">https://up.eparmaan.in</a></td>
<td>e-Pramaan specific</td>
</tr>
<tr>
<td>2*</td>
<td>SingleLogoutServiceURL</td>
<td>/processSLORequest.do</td>
<td>e-Pramaan specific</td>
</tr>
<tr>
<td>3*</td>
<td>SingleSignOnServiceURL</td>
<td>/processSSORequest.do</td>
<td>e-Pramaan specific</td>
</tr>
<tr>
<td>4</td>
<td>AssertionIssuer</td>
<td>e-Pramaan</td>
<td>For validation. This value will be string compared to validate the issuer is e-Pramaan</td>
</tr>
<tr>
<td>5</td>
<td>Issuer</td>
<td>Serviceld (Numeric value) given to the service by e-Pramaan when the service is registered at sp.eparmaan.in</td>
<td>Every service registered with e-Pramaan will be given a unique id called Serviceld.</td>
</tr>
<tr>
<td>6</td>
<td>SPServiceHomePageURL</td>
<td>Department Service has to enter this value</td>
<td>Base URL of the Service. Eg: <a href="http://dummypsp.in/localdemo1">http://dummypsp.in/localdemo1</a></td>
</tr>
<tr>
<td>7</td>
<td>SPAssertionConsumerServiceURL</td>
<td>/ssoresponseconsumer</td>
<td>Servlet which will receive SSO Response from e-Pramaan</td>
</tr>
<tr>
<td>8</td>
<td>SPLogoutConsumerURL</td>
<td>/logoutresponseconsumer</td>
<td>Page which will receive Logout Response from e-Pramaan</td>
</tr>
</tbody>
</table>
### Table 3.11 - URL Mapping at e-Pramaan

<table>
<thead>
<tr>
<th></th>
<th>SLOURL</th>
<th>URL of the SLO RESTful Web service. Used to validate SLO Request from e-Pramaan.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10*</td>
<td>EncryptionSeed</td>
<td>* This value will be shared with you by e-Pramaan. e-Pramaan mandates all request and responses be encrypted. This value will be given by e-Pramaan.</td>
</tr>
<tr>
<td>11*</td>
<td>EncryptionSalt</td>
<td>*Service ID of the service This is the Serviceld of the service. Refer item (5) above.</td>
</tr>
<tr>
<td>12</td>
<td>SamlSigning</td>
<td>True To enable/disable signing of SAML Request.</td>
</tr>
<tr>
<td>13</td>
<td>SPCertificateFilePath</td>
<td>Path to the private certificate of Service Private certificate of Department Service will be used for signing the SAML Request.</td>
</tr>
<tr>
<td>14</td>
<td>SPCertificatePassword</td>
<td>The private key of certificate is protected using this. This password is required for signing the SAML request/responses using Department Service’s private key.</td>
</tr>
</tbody>
</table>

* Marked values will be shared to the service by e-Pramaan administrator.

### 5. Register Department Service at e-Pramaan Service Portal

For integration with e-Pramaan, the URLs at Department Service have to be registered with e-Pramaan. The steps for service registration are as follows.

- Department Admin registers the department as a Service Provider(SP) on Service Portal of e-Pramaan.
- e-Pramaan admin gets notification of new SP registered on e-Pramaan.
- After receiving the request through proper channel, e-Pramaan admin activates the registered Department.
- SP admin logs in to e-Pramaan and adds new service. While adding service, (s)he enters the required URLs according to Table 3.11.
- e-Pramaan admin activates the newly registered service after testing of integration is completed.

Now, the Department Service is integrated with e-Pramaan & users can login using e-Pramaan to access the service.
<table>
<thead>
<tr>
<th>Sr. #</th>
<th>Field name at sp.epramaan.in</th>
<th>Value at SP portal</th>
<th>Value in SSO.properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Service Url</td>
<td>Service Url</td>
<td>[SPServiceHomePageURL]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Home page of Service</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ex: (<a href="http://dummysp.in/local">http://dummysp.in/local</a> demo1)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Logout Url</td>
<td>Logout Url</td>
<td>[SPServiceHomePageURL]+[SPLogoutConsumerURL]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>URL of the Servlet which consumes logout response sent by e-Pramaan Ex: (<a href="http://dummysp.in/local">http://dummysp.in/local</a> demo1/logoutresponseconsumer)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SSO Url</td>
<td>SSO Url</td>
<td>[SPServiceHomePageURL]+[S PAssertionConsumerService URL]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>URL of the Servlet which consumes logout response sent by e-Pramaan Ex: (<a href="http://dummysp.in/local">http://dummysp.in/local</a> demo1/ssoresponseconsumer)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SLO Url</td>
<td>SLO Url</td>
<td>Not mapped</td>
</tr>
<tr>
<td></td>
<td></td>
<td>URL of the REST Web Service which consumes SLO request sent by e-Pramaan Ex: (<a href="http://dummysp.in/local">http://dummysp.in/local</a> demo1/ws/saml/SLO)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Logout Failure URL</td>
<td>Logout Failure URL</td>
<td>Not mapped</td>
</tr>
<tr>
<td></td>
<td></td>
<td>URL of the REST Web Service which will receive Logout Response if Logout operation failed at e-Pramaan</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SSO Failure URL</td>
<td>SSO Failure URL</td>
<td>Not mapped</td>
</tr>
<tr>
<td></td>
<td></td>
<td>URL of the REST Web Service which will receive SAML Response if SSO operation failed at e-Pramaan</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>One time verification URL</td>
<td>One time verification URL</td>
<td>Not Mapped</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The one time verification URL must validate the user at the Service end and push the verification status to e-Pramaan REST WebService to complete the e-Pramaan User → Service User Id mapping.</td>
<td></td>
</tr>
</tbody>
</table>
6. Problems Faced During Integration

Table 6.1: Common problems faced during integration

<table>
<thead>
<tr>
<th></th>
<th>Common problems faced during integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SSO_MANAGE for session management – variable not found and related errors.</td>
</tr>
<tr>
<td>Soln.</td>
<td>Add the listener for session management as given below.</td>
</tr>
<tr>
<td></td>
<td>&lt;listener&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;listener-class&gt;in.cdac.epramaan.sp.samlconnector.SSOConnectorListener&lt;/listener-class&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/listener&gt;</td>
</tr>
<tr>
<td>2</td>
<td>PKIX certificate path construction failed error - related to e-Pramaan certificate issued by Indian CA</td>
</tr>
<tr>
<td>Soln.</td>
<td>Add the method to bypass the certificate path construction check. code given below.</td>
</tr>
<tr>
<td></td>
<td>This method must be called before the Rest Service is called.</td>
</tr>
<tr>
<td></td>
<td>public static void trustSelfSignedSSL()</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>try{</td>
</tr>
<tr>
<td></td>
<td>final SSLContext ctx = SSLContext.getInstance(&quot;TLS&quot;);</td>
</tr>
<tr>
<td></td>
<td>final X509TrustManager tm = new X509TrustManager();</td>
</tr>
<tr>
<td></td>
<td>@Override</td>
</tr>
<tr>
<td></td>
<td>public void checkClientTrusted(X509Certificate[] arg0, String arg1) throws CertificateException {</td>
</tr>
<tr>
<td></td>
<td>// TODO Auto-generated method stub</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>@Override</td>
</tr>
<tr>
<td></td>
<td>public void checkServerTrusted(X509Certificate[] arg0, String arg1) throws CertificateException {</td>
</tr>
<tr>
<td></td>
<td>// TODO Auto-generated method stub</td>
</tr>
</tbody>
</table>
3. **Could not write request: no suitable HttpMessageConverter found for request type**
[in.cdac.epramaan.sp.webservice.model.EnrolSPServiceResponseWrapper]

**Soln.** This error occurs because Jackson jar is missing in the spring project. Add the jar jackson-all-1.8.1.jar to the project to solve the issue.

4. **Error while creating SAML object - Parse error**

**Soln.** Verify these
- 1) They are using POST method.
- 2) They are posting to /auth/saml/processSSOrequest.do
- 3) They are sending service ID with SAML Request.

5. **Input length must be multiple of 16 when decrypting with padded cipher**

**Soln.** They are sending SAML request via GET without urlencoding the samlRequest. Ask them to change to POST method.

6. **Always authentication failure at e-Pramaan when using e-PramaanID based mapping**

**Soln.** e-Pramaan ID based mapping is yet to implement in https://up.epramaan.in. Ask them to use some other mapping.

7. **ISSUEINSTANT_VALIDATION_ERROR**

**Soln.** This error occurs because time difference between e-Pramaan server and SP server is more than 15 minutes. Kindly co-ordinate with the team and adjust the time(IST) so that difference is less than 15 minutes.
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8.</td>
<td><strong>How to get the SSOToken when e-Pramaan redirects to one time verification url?</strong></td>
</tr>
</tbody>
</table>
| Soln. | It is sent as parameter 'ssoToken'. Get the SSOToken by  
String encryptedSSOToken = request.getParameter("ssoToken"); |
| 9. | **java.security.InvalidKeyException: Illegal key size or default parameters** |
| Soln. | Because the unlimited strength cryptography policy files from oracle for the version jdk are missing. Download and copy the jars to JRE to solve the issue. |
| 10. | **java.lang.IllegalArgumentException: InputStream cannot be null**  
at javax.xml.parsers.DocumentBuilder.parse(Unknown Source)  
at org.opensaml.xml.parse.BasicParserPool$DocumentBuilderProxy.parse(BasicParserPool.java:696)  
at org.opensaml.xml.parse.BasicParserPool.parse(BasicParserPool.java:218)  
at org.opensaml.xml.XMLConfigurator.load(XMLConfigurator.java:142)  
at org.opensaml.DefaultBootstrap.initializeXMLTooling(DefaultBootstrap.java:224)  
at org.opensaml.DefaultBootstrap.initializeXMLTooling(DefaultBootstrap.java:207)  
at org.opensaml.DefaultBootstrap.bootstrap(DefaultBootstrap.java:100)  
at in.cdac.epramaan.sp.request.RequestManagerImpl.<clinit> |
<p>| Soln. | This error occurs in jdk 1.6 because xerces library is overriding the default xml parsing library. Remove the jar xercesImpl-2.10.0.jar from project to solve the issue. |
| 11. | <strong>Error: Unable to found KeyStore with the given keystore name</strong> |
| Soln. | Keystore file mentioned in the properties file is not found. Either place the properties file in the correct location or set SamlSigning = False in the configuration file. |
| 12. | <strong>SSOToken Manager jar not found</strong> |
| Soln. | Share epramaan-client.jar with the team. |
| 13. | <strong>Issue with ServiceUser user = (ServiceUser) request.getAttribute(&quot;user&quot;).</strong> |
| Soln. | This is dummy class created for testing internally. This must be replaced by their own logic. Take clues from OneTimeVerification.java for this. |</p>
<table>
<thead>
<tr>
<th></th>
<th>Question</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.</td>
<td>What is the push URL for onetime verification?</td>
<td><strong>Soln.</strong> <a href="https://up.epramaan.in/rest/epramaan/enrol/response">https://up.epramaan.in/rest/epramaan/enrol/response</a></td>
</tr>
<tr>
<td>15.</td>
<td><strong>ERROR:</strong> The method <code>getCredentialForSigning(Properties)</code> from the type <code>SPSamlUtil</code> refers to the missing type <code>Credential</code></td>
<td><strong>Soln.</strong> Include Opensaml library opensaml-2.6.4.jar &amp; openws-1.5.4.jar to the project.</td>
</tr>
</tbody>
</table>

**Contact us**
Centre for Development of Advanced Computing (C-DAC), Mumbai, 
Gulmohar Crossroad 9, Juhu, Mumbai 
Tel: +91-22-2620-1606/1574 
Fax: +91-22-26232195/ 26210139 
Email : epramaan@cdac.in