2024年11月8日

ユーザによる認証方式が選択できる設定

国立情報学研究所

教育研修事業

**目次**

[**1. はじめに 2**](#_heading=h.1fob9te)

[**2. 実習セミナーでは 2**](#_heading=h.3znysh7)

[**3. 手順書 3**](#_heading=h.tyjcwt)

[3.1「セキュリティレベルを設定したSPに対する認証」での設定確認 3](#_heading=h.rflhuw9q63nd)

[3.2 MFAフローの設定（ユーザによる認証方式が選択できる設定） 6](#_heading=h.cjbpse4praf9)

[● SPのセキュリティレベルに応じたフロー選択の設定 6](#_heading=h.70b724qhnmoc)

[● conf/authn/mfa-authn-config.xmlで各Levelに応じた認証設定 7](#_heading=h.dibjyprl732a)

[3.3 カスタムフローの設定 8](#_heading=h.3j8fnzwyl4xe)

[● methodChooserフローを定義するため、flows/methodChooser/ディレクトリを作成 8](#_heading=h.huw2vxdwmskk)

[● flows/methodChooser/methodChooser-flow.xmlファイルを作成して設定 8](#_heading=h.noj9c41zjd8k)

[● カスタムフロー画面の設定 9](#_heading=h.lqxqcftqcgke)

[**4. 動作確認 13**](#_heading=h.3dy6vkm)

# **1. はじめに**

※前提として、活用編メニュー「セキュリティレベルを設定したSPに対する認証」が実施済みであるとします。本メニューでは、IdPをカスタマイズします。

ログイン時に対象SPが使用できる認証方式をユーザが選択してログインできるようになります。セキュアなクライアント証明書認証などが行えるユーザは、ID/パスワードの入力が必要なく、SPにアクセスできます。

# **2. 実習セミナーでは**

以下のような設定で行います。

手順書と照らし合わせながら、作業を進めてください。

* モジュールの有効化

MultiFactor認証フローは、「IdPモジュール」と呼ばれる構成管理補助機能で管理され、MultiFactor認証フローのIdPモジュールは「idp.auth.MFA」となります。

Shibboleth IdP V5を新規インストールした場合、デフォルトで「idp.auth.MFA」は無効となっているため、有効化します。

* MFAフローの設定（ユーザによる認証方式が選択できる設定）

1. SPのセキュリティレベルに応じたフロー選択の設定を行います。
2. conf/authn/mfa-authn-config.xmlで各Levelに応じた認証設定を行います。

* カスタムフローの設定

1. カスタムフローを定義するため、flows/methodChooser/ディレクトリを作成します。
2. flows/methodChooser/methodChooser-flow.xmlファイルを作成して設定します。
3. views/chooser.vmファイルを作成して以下のように設定します。

# **3. 手順書**

以下は、MultiFactor認証フロー(MFA)を用いた認証設定のURLです。手順の詳細にご興味がある方はご参照ください。

[設定手順書](https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=26186832)

## 3.1「セキュリティレベルを設定したSPに対する認証」での設定確認

* /opt/shibboleth-idp/conf/authn/authn.properties の設定確認

idp.authn.flows にMFAを設定して、MFAフローを有効化されていることを確認してください。

|  |
| --- |
| # Regular expression matching login flows to enable, e.g. IPAddress|Password  #idp.authn.flows= Password  idp.authn.flows= MFA |

※ なお、idp.propertiesでもMFAを有効化する「idp.authn.flows = MFA」の記述が可能ですが、両ファイルでこれを設定すると次の警告メッセージがログに記録されます。混乱の元なので必ずauthn.propertiesで設定してください。

|  |
| --- |
| WARN - Ignoring duplicate property 'idp.authn.flows' |

* /opt/shibboleth-idp/conf/authn/authn.properties の設定確認

パスワード認証、クライアント認証、MFA認証フローの設定が以下の通り変更されていることを確認してください。

|  |
| --- |
| （省略）  Password認証がlevel1を満たすとする設定  idp.authn.Password.supportedPrincipals = \←#を外す  saml2/urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport, \←#を外す  saml2/urn:oasis:names:tc:SAML:2.0:ac:classes:Password, \←#を外す  saml2/urn:mace:gakunin.jp:idprivacy:ac:classes:Level1, \←挿入  saml1/urn:oasis:names:tc:SAML:1.0:am:password←#を外す  （省略）  クライアント証明書認証がlevel1, 2を満たすとする設定  <赤字の部分を挿入>  idp.authn.RemoteUser.checkHeaders = REMOTE\_USER  idp.authn.RemoteUser.supportedPrincipals = \  saml2/urn:oasis:names:tc:SAML:2.0:ac:classes:X509, \  saml2/urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient, \  saml2/urn:mace:gakunin.jp:idprivacy:ac:classes:Level2, \←挿入  saml2/urn:mace:gakunin.jp:idprivacy:ac:classes:Level1, \←挿入  saml1/urn:ietf:rfc:2246  （省略）  MFAフローがlevel1, 2を満たすとする設定  idp.authn.MFA.supportedPrincipals = \  saml2/urn:oasis:names:tc:SAML:2.0:ac:classes:InternetProtocol, \  saml2/urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport, \  saml2/urn:oasis:names:tc:SAML:2.0:ac:classes:Password, \  saml2/urn:mace:gakunin.jp:idprivacy:ac:classes:Level2, \←挿入  saml2/urn:mace:gakunin.jp:idprivacy:ac:classes:Level1, \←挿入  saml1/urn:oasis:names:tc:SAML:1.0:am:password  （省略） |

* /opt/shibboleth-idp/conf/relying-party.xml の設定確認

SAML2.SSOにdefaultAuthenticationMethodsプロパティが設定されていることを確認してください。

|  |
| --- |
| （省略）  <bean id="shibboleth.DefaultRelyingParty" parent="RelyingParty">  <property name="profileConfigurations">  <list>  <!-- SAML 1.1 and SAML 2.0 AttributeQuery are disabled by default. -->  <!--  <ref bean="Shibboleth.SSO" />  <ref bean="SAML1.AttributeQuery" />  <ref bean="SAML1.ArtifactResolution" />  -->  <bean parent="SAML2.SSO" p:postAuthenticationFlows="attribute-release">  <property name="defaultAuthenticationMethods">  <list>  <bean parent="shibboleth.SAML2AuthnContextClassRef"  c:classRef="urn:mace:gakunin.jp:idprivacy:ac:classes:Level1" />  </list>  </property>  </bean>  <ref bean="SAML2.ECP" />  <ref bean="SAML2.Logout" />  <!--  <ref bean="SAML2.AttributeQuery" />  -->  <ref bean="SAML2.ArtifactResolution" />  </list>  </property>  </bean>  （省略） |

* モジュールの有効化の確認

モジュール「idp.authn.MFA」が有効化されているか確認してください。

|  |
| --- |
| # /opt/shibboleth-idp/bin/module.sh -t idp.authn.MFA || /opt/shibboleth-idp/bin/module.sh -e idp.authn.MFA  INFO - Including auto-located properties in /opt/shibboleth-idp/bin/../conf/admin/admin.properties  INFO - Including auto-located properties in /opt/shibboleth-idp/bin/../conf/authn/authn.properties  INFO - Including auto-located properties in /opt/shibboleth-idp/bin/../conf/c14n/subject-c14n.properties  INFO - Including auto-located properties in /opt/shibboleth-idp/bin/../conf/services.properties  INFO - Including auto-located properties in /opt/shibboleth-idp/bin/../conf/saml-nameid.properties  INFO - Including auto-located properties in /opt/shibboleth-idp/bin/../conf/ldap.properties |

## 3.2 MFAフローの設定（ユーザによる認証方式が選択できる設定）

### **conf/authn/mfa-authn-config.xmlで各Levelに応じた認証設定**

以下のように/opt/shibboleth-idp/conf/authn/mfa-authn-config.xmlを変更します。

「methodChooser」に「Password」「RemoteUser」を設定します。

|  |
| --- |
| （省略）  <util:map id="shibboleth.authn.MFA.TransitionMap">  <!-- First rule calls a flow to display a view to select a method to run. -->  <entry key="">  <bean parent="shibboleth.authn.MFA.Transition" p:nextFlow="methodChooser" />  </entry>  <!-- Second rule decides what to call based on event signaled by the view. -->  <entry key="methodChooser">  <bean parent="shibboleth.authn.MFA.Transition">  <property name="nextFlowStrategyMap">  <map>  <!-- Maps event to a flow -->  <entry key="ChoosePassword" value="authn/Password" />  <entry key="ChooseRemoteUser" value="authn/RemoteUser" />  </map>  </property>  </bean>  </entry>  <!-- An implicit final rule will return whatever the final flow returns. -->  </util:map>  （省略） |

## 3.3 カスタムフローの設定

### methodChooserフローを定義するため、flows/methodChooser/ディレクトリを作成

|  |
| --- |
| # mkdir /opt/shibboleth-idp/flows/methodChooser |

### flows/methodChooser/methodChooser-flow.xmlファイルを作成して設定

以下のように/opt/shibboleth-idp/flow/methodChooser/methodChooser-flow.xmlを作成します。

※MFAフローの設定と同様に「Password」「RemoteUser」を設定します。

|  |
| --- |
| <flow xmlns="http://www.springframework.org/schema/webflow"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://www.springframework.org/schema/webflow http://www.springframework.org/schema/webflow/spring-webflow.xsd">    <!-- This is chooser of login flows. -->        <view-state id="DisplayChooserWebView" view="chooser">          <on-render>  <evaluate expression="environment" result="viewScope.environment" />  <evaluate expression="opensamlProfileRequestContext" result="viewScope.profileRequestContext" />  <evaluate expression="opensamlProfileRequestContext.getSubcontext(T(net.shibboleth.idp.authn.context.AuthenticationContext))" result="viewScope.authenticationContext" />              <evaluate expression="authenticationContext.getAvailableFlows().values().?[id matches 'authn/(Password|RemoteUser)']" result="viewScope.availableAuthenticationFlows" />  <evaluate expression="authenticationContext.getSubcontext(T(net.shibboleth.idp.ui.context.RelyingPartyUIContext))" result="viewScope.rpUIContext" />  <evaluate expression="T(net.shibboleth.utilities.java.support.codec.HTMLEncoder)" result="viewScope.encoder" />  <evaluate expression="flowRequestContext.getExternalContext().getNativeRequest()" result="viewScope.request" />  <evaluate expression="flowRequestContext.getExternalContext().getNativeResponse()" result="viewScope.response" />  <evaluate expression="flowRequestContext.getActiveFlow().getApplicationContext().containsBean('shibboleth.CustomViewContext') ? flowRequestContext.getActiveFlow().getApplicationContext().getBean('shibboleth.CustomViewContext') : null" result="viewScope.custom" />          </on-render>            <transition on="ChoosePassword"   to="ChoosePassword" />          <transition on="ChooseRemoteUser" to="ChooseRemoteUser" />      </view-state>        <end-state id="ChoosePassword" />      <end-state id="ChooseRemoteUser" />    </flow> |

### カスタムフロー画面の設定

methodChooserフローで使用する画面を作成します。以下のように/opt/shibboleth-idp/views/chooser.vmを作成します。

|  |
| --- |
| ##  ## Velocity Template for DisplayUsernamePasswordPage view-state  ##  ## Velocity context will contain the following properties  ## flowExecutionUrl - the form action location  ## flowRequestContext - the Spring Web Flow RequestContext  ## flowExecutionKey - the SWF execution key (this is built into the flowExecutionUrl)  ## profileRequestContext - root of context tree  ## authenticationContext - context with authentication request information  ## authenticationErrorContext - context with login error state  ## authenticationWarningContext - context with login warning state  ## ldapResponseContext - context with LDAP state (if using native LDAP)  ## username - username from previous rendering of form  ## errorMessageFunction - function to produce error message for form  ## rpContext - the context with information about the relying party (SP)  ## rpUIContext - the context with SP UI information from the metadata  ## encoder - HTMLEncoder class  ## cspDigester - Calculates base64-encoded SHA-2 hashes (call apply)  ## cspNonce - Calculates secure nonces (call generateIdentifier)  ## request - HttpServletRequest  ## response - HttpServletResponse  ## environment - Spring Environment object for property resolution  ## custom - arbitrary object injected by deployer  ##  #set ($rpContext = $profileRequestContext.getSubcontext('net.shibboleth.profile.context.RelyingPartyContext'))  ##  <!DOCTYPE html>  <html>  <head>  <title>#springMessageText("idp.title", "Web Login Service")</title>  <meta charset="UTF-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=5.0">  <link rel="stylesheet" type="text/css" href="$request.getContextPath()#springMessageText("idp.css", "/css/placeholder.css")">  </head>  <body>  <main class="main">  <header>  <img class="main-logo" src="$request.getContextPath()#springMessageText("idp.logo", "/images/placeholder-logo.png")" alt="#springMessageText("idp.logo.alt-text", "logo")" />    #set ($serviceName = $rpUIContext.serviceName)  #if ($serviceName && !$rpContext.getRelyingPartyId().contains($serviceName))  <h1>#springMessageText("idp.login.loginTo", "Login to") $encoder.encodeForHTML($serviceName)</h1>  #end  </header>    <section>  <form action="$flowExecutionUrl" method="post">  #parse("csrf/csrf.vm")    #\*  //  // SP Description & Logo (optional)  // These idpui lines will display added information (if available  // in the metadata) about the Service Provider (SP) that requested  // authentication. These idpui lines are "active" in this example  // (not commented out) - this extra SP info will be displayed.  // Remove or comment out these lines to stop the display of the  // added SP information.  //  \*#  #set ($logo = $rpUIContext.getLogo())  #if ($logo)  <img class="service-logo" src= "$encoder.encodeForHTMLAttribute($logo)" alt="$encoder.encodeForHTMLAttribute($serviceName)">  #end  #set ($desc = $rpUIContext.getServiceDescription())  #if ($desc)  <p>$encoder.encodeForHTML($desc)</p>  #end    #set ($errorMessage = $errorMessageFunction.apply($profileRequestContext))  #if ($errorMessage)  <p class="output-message output--error">$encoder.encodeForHTML($errorMessage)</p>  #end    <p><big>#springMessageText("idp.login.chooser", "Select Authentication.")</big></p>    <div class="grid">  <div class="grid-item">  #foreach ($authnFlow in $availableAuthenticationFlows)  #if ($authenticationContext.isAcceptable($authnFlow) and $authnFlow.test(profileRequestContext))  <button type="submit" name="\_eventId\_Choose$authnFlow.getId().replace('authn/','')">  #springMessageText("idp.login.$authnFlow.getId().replace('authn/','')", $authnFlow.getId().replace('authn/',''))  </button>  </div>  #end  #end  </div>  </div>    </form>    <ul>  <li><a href="#springMessageText("idp.url.helpdesk", '#')">#springMessageText("idp.login.needHelp", "Need Help?")</a></li>  </ul>  </section>  </main>  <footer class="footer">  <div class="cc">  <p>#springMessageText("idp.footer", "Insert your footer text here.")</p>  </div>  </footer>    </body>  </html> |

# **4. 動作確認**

※確認手順の説明に記載されている「動作確認用のSP」は、現在使用しているフェデレーションによってアクセス先が変わります。（どちらかのフェデレーションに参加して利用できる状態にしておいてください。）

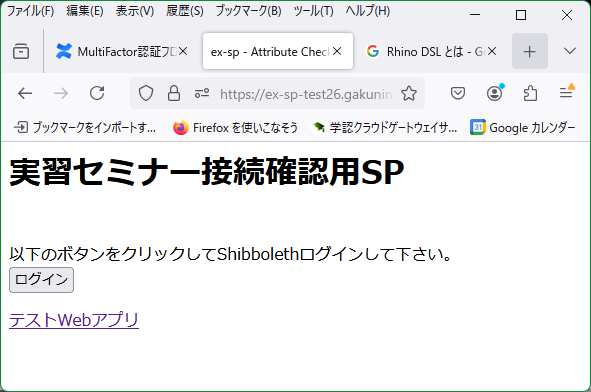
|  |  |
| --- | --- |
| 実習セミナーフェデレーション | <https://ex-sp.gakunin.nii.ac.jp/> |
| テストフェデレーション | <https://test-sp1.gakunin.nii.ac.jp/> |

① 設定後、Jettyの再起動を行ってない場合は行ってください。

|  |
| --- |
| systemctl restart jetty |

② 各自が使用するSPの接続確認用ページにアクセスします。

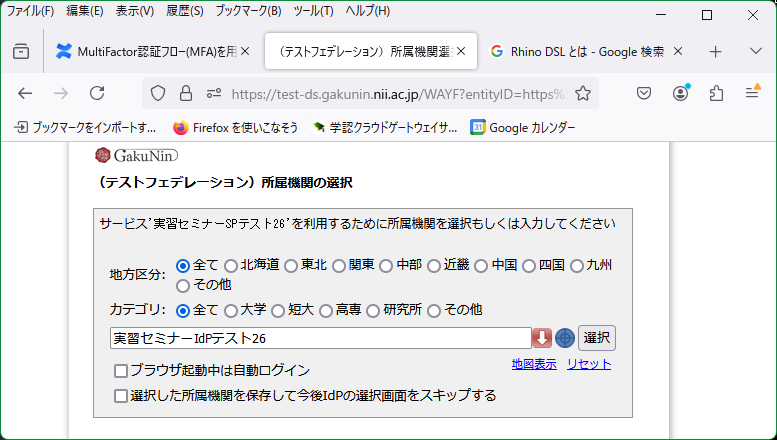
|  |
| --- |
| 例）1番を割り振られた場合  　https://ex-sp-test01.gakunin.nii.ac.jp/ |



#### **図　各自が使用するSPの接続確認用ページ**

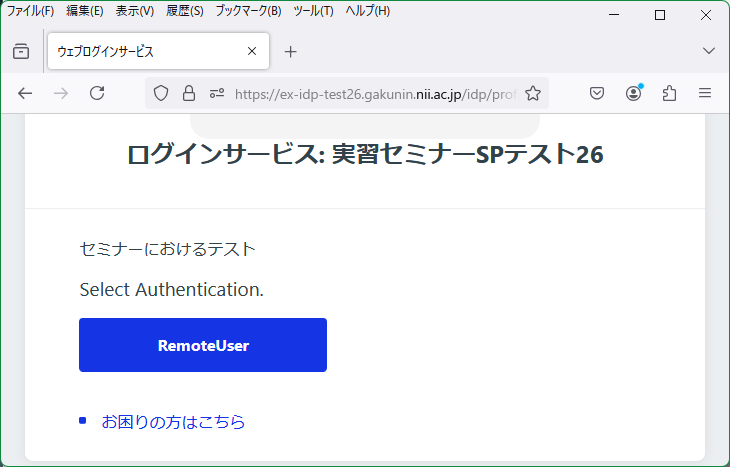
③ ログインボタンをクリックします。

④ DSの設定を行っている場合、所属機関の選択画面が表示されるので、各自が使用するIdPを選択します。



#### **図　各自が使用するIdP選択画面**

⑤ 認証方式を選択する画面が表示されるのですが、各自が使用するSPのセキュリティレベルが証明書認証を必要と設定しているので、RemoteUserボタン（クライアント証明書）のみ表示され、選択可能となります。



#### 図　RemoteUserボタンのみ表示された画面

⑥ 選択すると個人証明書の要求というダイアログが表示されるので、対象となるクライアント証明書を選択して、OKボタンをクリックします。

※送信属性同意画面が表示される場合は、そのまま設定値を送信します。

⑦ 正しく属性受信の確認ページが表示される事を確認してください。



#### 図　属性受信確認ページ

⑧ この状態で上述の動作確認用のSPにアクセスすると、セキュリティレベルが低いためSSOにより認証がスキップされます。

「接続テスト」ボタンをクリックし、IdPとして「実習セミナーIdPテストXX」を選択すると、パスワード等の入力なしで、正しく属性受信の確認ページが表示されます。



#### **図　動作確認用のSPの画面**



#### **図　「実習セミナーIdPテストXX」の選択**

⑨ 一度ブラウザを閉じて、動作確認用のSPにアクセスします。

⑩ 進めていくと動作確認用のSPはID/パスワード認証でも認証が可能であるため、認証方式を選択する画面では、Passwordボタン（ID/パスワード）とRemoteUserボタン（クライアント証明書）の２つのボタンが表示されます。



#### **図　認証方式が２つ表示された画面**

⑪ Passwordボタン（ID/パスワード）を選択した場合は、次に遷移するページでユーザ名とパスワードを入力してLoginボタンをクリックしてください。



#### **図　Password認証方式を選択した画面**

⑫ 認証後、正しく属性受信の確認ページが表示される事を確認してください。

⑬ この状態で各自が使用するSPにアクセスすると、セキュリティレベルが上なので⑤と同じ画面が表示されます。