Chapter 8. SAML Setup for i2b2 (v1.7.13 release)

Overview

i2b2 now includes support for SAML-based enterprise authentication via an institutional Identity Provider. This guide walks through the webclient, user, and server configuration. This should be done after both server and client install.

> *This guide uses* **sp.example.org** *as the domain name. Please replace* **sp.example.org** *with your domain name.*

Prerequisites

The following applications and services must be already setup and running:

- i2b2 core server 1.7.13 release
- i2b2 web client release 1.7.13 (using the Apache web server)
- i2b2 database

We recommend all software frameworks are updated to the latest supported version:

- Wildfly 17.0.1 Final
- Apache HTTPD 2.0 or 2.2
- Apache Axis 2: 1.7.1
- PHP: >= 7.2.27

Requirements

- Administrative privileges.

Install Shibboleth (SAML Service Provider) and AJP

>

> *The specific commands for installing packages are for CentOS 7. Commands might be slightly different in other versions and flavors of Linux.*

Preparing for Software Installation

Updating the Operating System.

It is generally best practice to update the operating system to get the latest security patches and software updates before installing any new software.

Execute the following command to update the operating system:

```
sudo yum -y update
```

Restart the server for the changes to apply.

Installing EPEL (Extra Packages for Enterprise Linux)

Please visit Extra Packages for Enterprise Linux for more information.

Execute the following command to install additional open source packages:

```
sudo yum -y install epel-release
```

Run update again to pull the packages:

```
sudo yum -y update
```

Installing Shibboleth

Installing Shibboleth Service Provider (SP)

Add Shibboleth repository:

```
sudo wget <http://download.opensuse.org/repositories/security://shibboleth/CentOS_7/security:shibboleth.repo> \
-P /etc/yum.repos.d
```

Update the repository:

```
sudo yum -y update
```

Install Shibboleth:

```
sudo yum -y install shibboleth
```

Enable Shibboleth and restart Apache HTTP server:

```
sudo systemctl enable shibd
sudo systemctl start shibd
sudo systemctl restart httpd
```

Verifying Installation

Verify that Shibboleth has been properly installed.

Confirm Shibboleth functionality:

```
sudo shibd -t
```

You should see output response that ends with `overall configuration is loadable, check console or log for non-fatal problems`

Confirm Apache functionality:

```
sudo apachectl configtest
```

You should see the output `Syntax OK`.

Confirm shibd functionality:

Open up a web browser and navigate to https://sp.example.org/Shibboleth.sso/Session

> *Note: replace* sp.example.org *with your domain name.*

You should see the message **A valid session was not found.** in your browser.

Configuring Shibboleth

Configuring the Apache HTTP Server

Modify the ***shib.conf*** located in the directory **/etc/httpd/conf.d**.

Delete the following configuration:

```
<Location /secure>
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require shib-session
</Location>
```

Add the following configuration:

```
<Location />
  AuthType shibboleth
  ShibRequestSetting requireSession 0
  require shibboleth
</Location>
```

Setting Up Federation Files and Metadata

Your institution should provide you the IdP metadata to register your application or service with their Identity Provider (IdP). For an example, the Harvard University Information Technology (HUIT) provides a guide and files to register with their IdP: https://iam.harvard.edu/resources/saml-shibboleth-integration.

Updating the Shibboleth2 XML File

Modify the ***/etc/shibboleth/shibboleth2.xml*** file.

Update the SP Entity ID:

Modify the attributes of the **ApplicationDefaults** element as follows:

```
<!-- The ApplicationDefaults element is where most of Shibboleth's SAML bits are defined. -->
<ApplicationDefaults entityID="<https://sp.example.org/shibboleth">
                        REMOTE_USER="eduPersonPrincipalName,eppn"
                        cipherSuites="DEFAULT:!EXP:!LOW:!aNULL:!eNULL:!DES:!IDEA:!SEED:!RC4:!3DES:!kRSA:!SSLv2:!
SSLv3:!TLSv1:!TLSv1.1"
                        signing="true"
                        attributePrefix="AJP_">
```

> *Remember to replace* sp.example.org *with your domain name.*

Set the IdP Entity ID:

Modify the `<SSO>` tag as follow:

```
<SSO entityID="<https://example.org/saml2/idp/metadata.php>">
    SAML2
</SSO>
```

> *Remember to replace https://example.org/saml2/idp/metadata.php with your IdP entity.*

Modify the `<Handler>` Tags:

Replace the following:

```
<!-- Extension service that generates "approximate" metadata based on SP configuration. -->
<Handler type="MetadataGenerator" Location="/Metadata" signing="true"/>

<!-- Session diagnostic service. -->
<Handler type="Session" Location="/Session" showAttributeValues="true"/>
```

Point to the IdP Metadata:

The IdP metadata file should be placed in the directory **/etc/shibboleth**. In this example, the IdP metadata file is ***/etc/shibboleth/federation-metadata.xml***

Add the following:

```
<!-- Example of locally maintained metadata. -->
<MetadataProvider type="XML" validate="true" path="federation-metadata.xml"/>
```

> *Remember to replace* **federation-metadata.xml** *with the name of your IdP metadata file located in the directory /etc/shibboleth.*

Updating the Attribute-Map XML File

The ***attribute-map.xml***, located in the directory **/etc/shibboleth**, contains the names of the SAML 2.0 attributes that can be mapped to the IdP attributes.

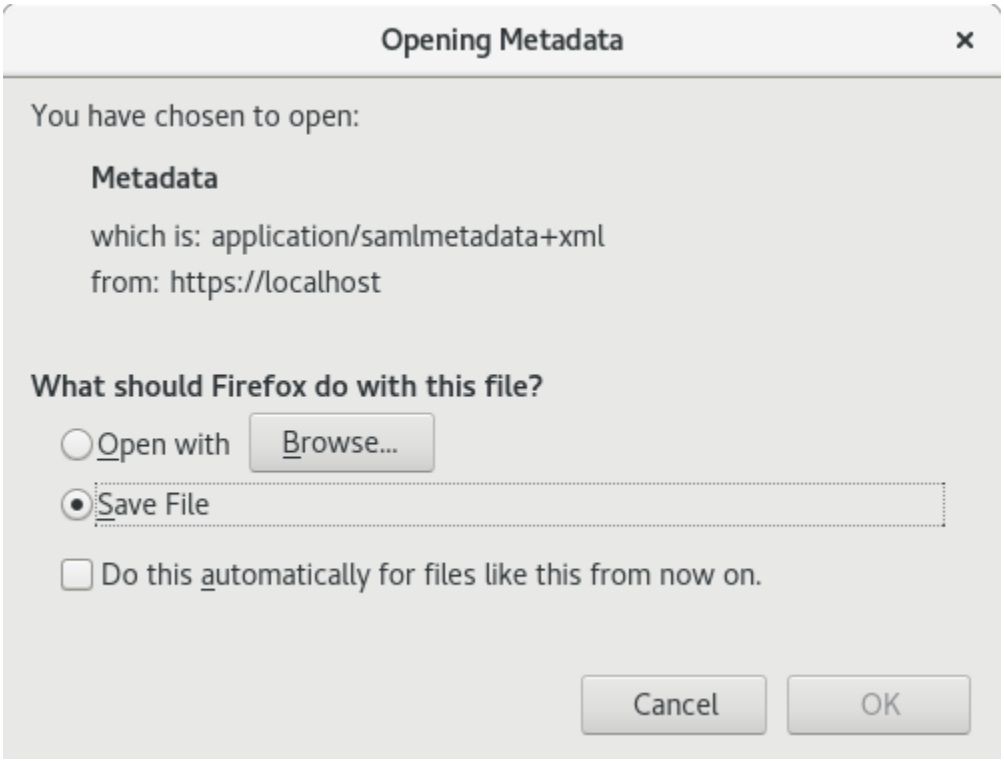
Add the following attribute mapping to the file ***/etc/shibboleth/attribute-map.xml***.

```
<Attribute name="uid" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" id="uid"/>
<Attribute name="eduPersonPrincipalName" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" id="
eduPersonPrincipalName"/>
<Attribute name="eduPersonAffiliation" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" id="
eduPersonAffiliation"/>
<Attribute name="mail" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" id="mail"/>
<Attribute name="displayName" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" id="displayName"/>
<Attribute name="givenName" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" id="givenName"/>
<Attribute name="sn" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" id="sn"/>
```

> *Note that your IdP attributes may be different. Please change the attribute* ***name*** *to the correct IdP attribute name. Do* ***NOT*** *change the* ***id*** *attribute.*

Getting the Service Provider Metadata

Open up a web browser and navigate to https://sp.example.org/Shibboleth.sso/Metadata. You should see a dialog for opening the metadata file ***Metadata***. Instead of opening it up, download it onto your computer.



> *You can rename the file Metadata to Metadata.xml for readability.*

The metadata file contains information about the Service Provider (SP) including the entity ID and the public certificates for signing and encryption. Regisiter this file with your Identity Provider (IdP).

> *Note that the signing certificate and encryption certificate included in the metadata file are from* **/etc/shibboleth/sp-signing-cert.pem** *and* **/etc/shibboleth/sp-encrypt-cert.pem**, *respectively. If you want to use your own certificates, just replace them along with the private keys and regenerate the metadata.*

Configuring the Apache JServ Protocol (AJP)

With the latest version of i2b2, the request to the Hive is no longer made directly to Wilfly. Instead, the request is made to the Apache HTTP server and then gets proxied over to Wildfly via AJP.

AJP is a trusted protocol and should never be exposed to untrusted clients. The communication between the clients is insecure (data is sent in clear text) and assumes that your network is safe. The configuration below will prevent AJP from being exposed.

Protect the AJP Connection With a Secret

Create a secret key to used by the Apache HTTP server and Wildfly. You can generate a random key here https://randomkeygen.com/ .

For the purpose of this guide, we will use ***5F6C696F56D37BCFD1296C3E33A11*** as the secret key.

Configuring the Apache HTTP Server

Restricting Port 80 to Only to Localhost

Modify the file ***httpd.conf*** located in the directory **/etc/httpd/conf/** to restrict Apache to listen to port 80 only to IP 127.0.0.1 (localhost):

```
# Listen 80
Listen 127.0.0.1:80
```

Adding AJP Configuration

Create a file named ***ajp.conf*** in the directory **/etc/httpd/conf.d/** with the following content, depending on what version of CentOS or Apache you are running:

CentOS 6 / Apache 2.0	**CentOS 7 / Apache 2.2 (recommended)**	**CentOS 8**
<VirtualHost *:80> <Location /i2b2/> ProxyPass ajp://localhost:8009/i2b2/ secret=5F6C696F56D37BCFD1296C3E33A11 ProxyPassReverse ajp://localhost:8009/i2b2/ secret=5F6C696F56D37BCFD1296C3E33A11 </Location> </VirtualHost>	<VirtualHost 127.0.0.1:80> ProxyRequests Off ProxyPreserveHost Off <Location /i2b2/services/> Require ip 127.0.0.1 ProxyPass ajp://localhost:8009/i2b2/services/ secret=5F6C696F56D37BCFD1296C3E33A11 </Location> </VirtualHost>	Not Supported

> *Remember to replace the secret key* 5F6C696F56D37BCFD1296C3E33A11 *with your own.*

Configuring Wildfly

Modify the file ***standalone.xml*** in the directory **/opt/wildfly/standalone/configuration** to enable AJP:

Ensure that AJP port is enable and set to *8009*:

```
<socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">
    ...
    ...
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    ...
    ...
</socket-binding-group>
```

Set AJP listener and secret key:

```
<subsystem xmlns="urn:jboss:domain:undertow:9.0" default-server="default-server" default-virtual-host="default-
host" default-servlet-container="default" default-security-domain="other" statistics-enabled="${wildfly.undertow.
statistics-enabled:${wildfly.statistics-enabled:false}}">
    <buffer-cache name="default"/>
    <server name="default-server">
        <ajp-listener name="ajp" socket-binding="ajp" max-post-size="10485760000" scheme="http"/>
        ...
        ...
        <host name="default-host" alias="localhost">
            ...
            <filter-ref name="secret-checker" predicate="equals(%p, 8009)"/>
            ...
        </host>
    </server>
    ...
    ...
    <filters>
        <expression-filter name="secret-checker" expression="not equals(%{r,secret},
'5F6C696F56D37BCFD1296C3E33A11') -> response-code(403)"/>
    </filters>
</subsystem>
```

> *Remember to replace the secret key* 5F6C696F56D37BCFD1296C3E33A11 *with your own.*

Restarting the Servers

Execute the command below to restart Apache HTTP server and Wildfly:

```
sudo systemctl restart wildfly.service
sudo systemctl restart httpd.service
```

Updating i2b2 Applications

Updating i2b2 Core Server

Stop Wildfly:

```
systemctl stop wildfly.service
```

Download the latest i2b2 core server from Github: https://github.com/i2b2/i2b2-core-server

Configure, compile, and install the latest version. For more information, please see the installation guide

Start Wildfly:

```
systemctl start wildfly.service
```

Updating i2b2 Webclient

Modify the ***i2b2_config_data.js*** file located in the directory **/var/www/html/webclient/** as needed. In particular, you will want a domain entry with `loginType = "federated"` (see *1.4.2 Domain Configuration*) and you will want to update the URL to begin with 127.0.0.1 (to use the AJP proxy to the Hive).

```
{
    urlProxy: "index.php",
    urlFramework: "js-i2b2/",
    startZoomed: true,
    lstDomains": [
        {
            domain: "i2b2demo",
            name: "HarvardDemo SAML",
            allowAnalysis: true,
            urlCellPM: "http://127.0.0.1/i2b2/services/PMService/",
            registrationMethod: "saml",
            loginType: "federated",
            showRegistration: true,
            installer: "/webclient/plugin_installer/",
            debug: true
        },
        {
            domain: "i2b2demo",
            name: "HarvardDemo",
            allowAnalysis: true,
            urlCellPM: "http://127.0.0.1:9090/i2b2/services/PMService/",
            registrationMethod: "",
            loginType: "local",
            showRegistration: false,
            debug: true
        }
    ]
}
```

Optionally, you can follow these directions to customize the SAML IdP login button:
i2b2 SAML: Customize Identity Provider (IdP) Login Button

Restart the Apache web server:
`systemctl restart httpd.service`

Updating the Lookup Table in the Database

Run the following SQL queries to update the pm_cell_data table. This will change your URL to begin with 127.0.0.1, which will use the AJP proxy. Note that your specific URL could be different depending on configuration.

PostgreSQL:

```
UPDATE pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/QueryToolService/'> WHERE cell_id  = 'CRC';
UPDATE pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/FRService/'> WHERE cell_id  = 'FRC';
UPDATE pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/OntologyService/'> WHERE cell_id  = 'ONT';
UPDATE pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/WorkplaceService/'> WHERE cell_id  = 'WORK';
UPDATE pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/IMService/'> WHERE cell_id  = 'IM';
```

Oracle:

```
UPDATE i2b2pm.pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/QueryToolService/'> WHERE cell_id  = 'CRC';
UPDATE i2b2pm.pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/FRService/'> WHERE cell_id  = 'FRC';
UPDATE i2b2pm.pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/OntologyService/'> WHERE cell_id  = 'ONT';
UPDATE i2b2pm.pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/WorkplaceService/'> WHERE cell_id  = 'WORK';
UPDATE i2b2pm.pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/IMService/'> WHERE cell_id  = 'IM';
```

SQL Server:

```
UPDATE i2b2pm.dbo.pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/QueryToolService/'> WHERE cell_id  =
'CRC';
UPDATE i2b2pm.dbo.pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/FRService/'> WHERE cell_id  = 'FRC';
UPDATE i2b2pm.dbo.pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/OntologyService/'> WHERE cell_id  =
'ONT';
UPDATE i2b2pm.dbo.pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/WorkplaceService/'> WHERE cell_id  =
'WORK';
UPDATE i2b2pm.dbo.pm_cell_data SET url = '<http://127.0.0.1/i2b2/services/IMService/'> WHERE cell_id  = 'IM';
```

> *Note that your database schema may be different.*

Configure users for SAML authentication

1. Create a user for each SAML-authenticating user with the user name set to the SAML EPPN.
2. Create a pm_user_param setting authentication_method to SAML.
3. Alternately, configure the user registration tool for federated mode, and it will allow users to create an i2b2 account if they have a valid SAML authentication. (Such users will still not have access to any projects until the administrator adds projects to his/her profile.)

Frequently Asked Question (FAQ)

Apache JServ Protocol (AJP)

AJP is a binary protocol that is used to proxy requests from a web server to an application server. It is a highly trusted protocol and should only be exposed to trusted clients.

How is AJP Used?

For security purposes, it is highly recommended to put the Wildfly server (application server) behind the Apache web server on the same network. The Apache web server should be configured to only accept incoming requests on port 443 (https), therefore, blocking direct access to Wildfly on port 9090 via http.

All inbound request to Wildfly will go through the Apache web server on port 443 (https). The web server then prox the requests over to Wildfly via AJP. With this setup, i2b2 can limit what method to expose to the clients outside the network.

Why Use AJP Instead of HTTP?

AJP has performance advantage over HTTP. HTTP is plain-text protocol, making it quite expensive in terms of band width. AJP carries the same information as http but in a more compact format. With AJP, request method can be reduced to a single byte and the header can be reduced to two bytes. For more information, please see the "Basic Packet Structure" section of the Apache Module mod_proxy_ajp documentation.

> *Note that AJP is **not required** to run i2b2. It is recommended for SAML authentication setup.*

Shibboleth

Shibboleth Variables in i2b2

All Shibboleth variable attributes must be prefixed with "**AJP**_".

What Shibboleth variables are required?

Below is a list of varibles that are required by the webclient.

Variable	Type
uid	User ID
eduPersonPrincipalName	User's Identifier
eduPersonAffiliation	User's Organization
mail	User's Email
displayName	User's Display Name
givenName	User's First Name
sn	User's Last Name

How are Shibboleth Variables Passed From Apache to Wildfly ?

When the user clicks the "Sign In" button, the request does not go directly to the i2b2 PM cell. Instead, the request goes back to the webclient server for further processing and then proxied over to the PM cell. This process occurs in all version of i2b2.

After the user sign with his/her organization's login credentials, the IdP redirects back to the webclien along with the Shibboleth variables. The webclient automatically submit a login request with ***eduPersonPrincipalName*** as the user name and Shibbleth's session ID as the password. As mentioned above, the request does not go directly to the PM cell. The request goes back to the webclient server for further processing. At this time, the Shibboleth variables are set as the Apache environment variables. The Shibboleth variable attributes and values are placed in the request headers send to the PM cell. The PM cell extracts the eduPersonPrincipalName and the shibboleth session ID from the headers and compares them with the ones submitted in the login form for validation.

The process where the webclient sends the Shibboleth variables in the request headers to the PM cell must be secured. Therefore, it is recommended not to allow direct access to the PM cell outside the network. It is a good practice to have the PM cell sits behind the Apache server and have all requests from the webclient proxy over the PM cell via AJP. ***Note that this is not an issue when the webclient is not setup to use SAML authentication.***

> *Note that Shibboleth is not required to run i2b2 when SAML is not used for authentication.*

i2b2 Authetication Flow

Note: The original version of the setup guide is here: https://github.com/kvb2univpitt/i2b2-saml-demo/tree/main/doc/saml_setup