Install Guide for i2b2 Web Client v1.8 on Apache

1. i2b2 Web Client Install · https://i2b2.org/docs/install-guide-for-i2b2-web-client-v1-8-on-apache/

Unknown macro: 'projectdoc-box-caution'

A-hat character

If obfuscated users are seeing an odd A-hat character in query results, UTF-8 needs to be enabled in Apache. (In the charset.conf configuration file, find the line that reads "AddDefaultCharset UTF-8". Uncomment it and restart Apache.)

See detailed instructions in Troubleshooting.

PHP "display_errors"

If login does not work, check that PHP is not echoing warnings into the XML response. Make sure " display_errors = Off" is in your php.ini!

WEB CLIENT HOSTING VIA APACHE

General Hosting Installation

In general, you will need to copy all the files and directories into the root hosting directory that you have configured for your web server software. The majority of the web client's files will be served directly by the web server. Certain functionality (primarily the proxy service and SAML-related services) are enabled using PHP script files. To work properly within Apache, you will need to configure the Apache service to execute the PHP scripts contained within the hosting directory. If you are using the NodeJS-based proxy server these things are self-contained.

Requirement for a Proxy Service

The i2b2 server architecture hosts all i2b2 services within a Java application server JVM. No services within the i2b2 hive exist to host the HTML, CSS and Javascript files that make up the i2b2 web client. To accomplish this the web client's files must be hosted somewhere. Because of the "same-origin" security rule that is baked into all web browsers, the i2b2 web client is not able to directly connect to the different URLs which expose the i2b2 services. In order to accomplish this a "proxy service" must be operated within the web server environment which is hosting the i2b2 web client's code. When hosting the web client using Apache the proxy service is manifested via the **proxy.php** script file. If you are hosting the web client using the NodeJS-based proxy server the proxy service is self-contained within its code.

Requirement for Whitelist

Whether hosting via NodeJS or Apache, the proxy service uses a whitelist to prevent it from being used in relay attacks to subvert network security. A whitelist is a list of domain names or IP addresses that the proxy will *allow* requests to be forwarded to.

Whitelist in Apache

To edit the whitelist when the web client is hosted in Apache you will need to edit the values within the **proxy.php** file. At the top of the file will be a PHP variable called **$WHITELIST** which will contain an array of URLs. You will need to enter the protocol, hostname and port (if any) for all i2b2 hive services that the web client will need access to. You can see an example of a proper whitelist configuration below:

```
$WHITELIST = array(
 "http://services.i2b2.org",
 "http://127.0.0.1:9090",
 "http://127.0.0.1:8080",
 "http://127.0.0.1",
 "http://localhost:8080",
 "http://localhost:9090",
 "http://localhost"
 );
```

Blacklist in Apache

In the PHP proxy service file (proxy.php) you will also have the ability to set blacklist entries. These entries will prevent proxying of any requests whose URLs begin with any of the values listed in the **$BLACKLIST** variable. You can see an example of a proper blacklist configuration below:

```
$BLACKLIST = array(
"http://127.0.0.1:9090/test",
"http://localhost:9090/test"
);
```

Security Check Each i2b2 Request in Apache

It is possible for you in Apache to run a security check against the Project Manager for each i2b2 request. These options are within the **proxy.php** file as follows:

```
$pmURL = "http://127.0.0.1:8080/i2b2/rest/PMService/getServices";
$pmCheckAllRequests = true;
```

The **$pmURL** variable should be set to the URL used to validate the security credentials in the request before it is forwarded on to the requested cell service. It usually ends in "/getServices". The **$pmCheckAllRequests** variable is used to turn on or turn off the extra per-request security check. It is a boolean value and should be set to true or false only.

SAML AUTHENTICATION

Overview of SAML Authentication

Security Assertion Markup Language, or SAML, is a standardized way to tell external applications and services that a user is who they say they are. SAML makes single sign-on (SSO) technology possible by providing a way to authenticate a user once and then communicate that authentication to multiple applications. SAML authentication functionality was added in i2b2 version 1.7.8 and later. If you need users to log into i2b2 using Shibboleth that means that you will enable and configure SAML

authentication on your i2b2 hive.

SAML Authentication via Apache with PHP

When hosting the i2b2 web client under Apache, SAML authentication services are managed via the following files: **saml-acs.php** (for redirection to SAML authentication server), **saml-redirect.php** (when browser is returning from SAML authentication server) and **saml-logout.php** (used to clear credential cookies during logout). You will also need to set up Apache for SAML using Shibboleth (which is beyond the scope of this document). Documentation on this is in Chapter 8 - "*SAML Setup for i2b2*" in the "*i2b2 Installation Guide*" found at {+} https://i2b2.atlassian.net/wiki/display/getstarted/i2b2+Installation+Guide+.

Web Client Configuration for SAML Authentication

In addition to configuring the web server and proxy service you will need to configure the i2b2 domain instance within the web client. Once again, all i2b2 domains are configured within the **i2b2_config_domains.json** file. Specifically you will need to add a "saml" object within the SAML-enabled domain entry. Within the saml object will be another object named with the name of the SAML-enabled service (for example "shibboleth" as shown below). When you are using the NodeJS-based proxy server you can set the value of the object to true or null. When hosting your web client deployment under Apache you will need to specify the URLs for the redirect and logout services within the saml object. Example configurations are as follows:

```
{
  "urlProxy": "proxy.php",
  "urlFramework": "js-i2b2/",
  "lstDomains": [
    {
      "domain": "i2b2demo",
      "name": "Shibboleth (localhost)",
      "urlCellPM": "http://127.0.0.1/i2b2/services/PMService/",
      "allowAnalysis": true,
      "debug": false,
      "saml" : {
         "shibboleth": {
```

```
            "redirect": "saml-redirect.php",
            "logout": "saml-logout.php"
        }
      }
   }
 ]
}
```

I2B2 CONFIGURATION

Configuring Proxy Service URL

When the i2b2 web client is loaded into a web browser it loads the configuration file **i2b2_config_domains.js** located in the root of the hosting directory. This file contains (among other things discussed below) an entry called **urlProxy** that configures the URL that the web client will use to access the proxy service to communicate with the i2b2 hive. By default the is configured for self-hosting using the NodeJS i2b2-webclient-proxy server. To host the web client under Apache you will need to change this from the default as shown below:.

For hosting via NodeJS

```
{
   "urlProxy": "/~proxy",
   "urlFramework": "js-i2b2/",
   "lstDomains": [...]
}
```

For hosting via Apache + PHP

```
{
   "urlProxy": "/proxy.php",
   "urlFramework": "js-i2b2/",
   "lstDomains": [...]
}
```

Configuring the Timeline

In order to use legacy plugins like the timeline, you will also have to update the **/js-i2b2/cells/LEGACYPLUGIN/legacy_plugin/i2b2_config_data.js** file following the same pattern. Additionally, copy /proxy.php to the **/js-i2b2/cells/LEGACYPLUGIN/** so that it is accessible to the legacy plugins.

Configuring Connections to i2b2 Servers

All configurations to connect to i2b2 servers are stored within the configuration file **i2b2_config_domains.js** and are contained within the **lstDomains** array. By default a connection to the demonstration server at http://services.i2b2.org is configured for testing purposes and is shown below:

```
{
 "urlProxy": "/proxy.php",
 "urlFramework": "js-i2b2/",
 "lstDomains": [
   {
     "domain": "i2b2demo",
     "name": "i2b2.org Demo",
     "urlCellPM": "http://services.i2b2.org/i2b2/services/PMService/",
     "allowAnalysis": true,
     "debug": false
   },
   {...},
   {...},
 ]
}
```

The **domain** value contains the domain name that is configured within an i2b2 instance.
The **name** value contains the text that is displayed in the login window for the user to identify the instance.
The **urlCellPM** value contains the URL that is used to contact the i2b2 server which contains the configured domain. The hostname of the server must also be added to the whitelist configuration

used by the proxy service. See the related whitelist documentation in the NodeJS/Apache Hosting section.

The **allowAnalysis** boolean value configures whether a user will be able to use analysis plugins when logged into the i2b2 domain.

The **debug** boolean value configures whether the user will be able to use debugging tools when logged in. This value is deprecated and will be removed in the future.

PLUGIN OVERVIEW

Modern Plugins

All "modern" style plugins (release version 1.8 and later) will be loaded and operate inside of their own HTML iframe within the larger i2b2 web client. They will be allowed to use any and all Javascript libraries that they wish (ex. jQuery, D3js, React, Angular, Vue, etc). All modern-style plugins are placed in the **plugins** directory within the root of the web client hosting directory. Each plugin is placed within its own directory according to its code-wise name.

Legacy Plugins

One of the accomplishments of the i2b2 web client v1.8 rewrite was the preservation of backwards compatibility with previously written plugins. In order to do this, a specialty environment was created which exists within an iframe of the modern i2b2 web client UI. Within this iframe, legacy plugins are added to the environment like it is simply a stand-alone instance of the older (pre-v1.8) web client software. The legacy execution environment exists at **/js-i2b2/cells/LEGACYPLUGIN/legacy_plugin/**.

CONFIGURATION OF MODERN PLUGINS

File Locations

All plugins will exist entirely within the **/plugins** directory. Each plugin will be hosted in a single directory that follows Java-style library paths such as **/edu/harvard/catalyst/example** with the resulting plugin name being defined as "edu.harvard.catalyst.example". All files and libraries needed by the plugin will be contained solely within its own directory.

Configuration File for Plugin System

For plugins to operate properly when hosted by Apache you will need to manually edit a file called **plugins.json** that exists within the root **/plugins** directory containing an array of plugin names as shown below.

```
[
  "edu.harvard.catalyst.example",
  "edu.harvard.WeberLab.ExportPatientset"
]
```

If you are using the i2b2-webclient-proxy service to host your web client UI it will automatically generate the **plugins.json** file based on the directory structure if the file does not exist (not manually created). The Github repository for this proxy server is at _https://github.com/hms-dbmi/i2b2-webclient-proxy_

CONFIGURATION OF LEGACY PLUGINS

File Locations

The execution environment for legacy plugins exists within the LEGACYPLUGIN cell of the modern web client. All execution files will exist within subdirectories of this cell. The cell's full directory is **/js-i2b2/cells/LEGACYPLUGIN/** and the execution environment for legacy plugins exists at **/js-i2b2/cells/LEGACYPLUGIN/legacy_plugin/**. Any legacy plugins that need to be installed would be saved in the **plugins** directory under the legacy environment's plugin directory which expands to the full path of: **/js-i2b2/cells/LEGACYPLUGIN/legacy_plugin/js-i2b2/cells/plugins/**.

Registering Legacy Configuration Files

Just like in the legacy web client, any plugins that are added must have an entry within the **i2b2_loader.js** file that is in the root of the execution environment. This file would be located at **/js-i2b2/cells/LEGACYPLUGIN/legacy_plugin/js-i2b2/i2b2_loader.js** with an example listing of the configuration lines shown below:

```
//================================================================================//
// THESE ARE ALL THE CELLS THAT ARE INSTALLED ONTO THE SERVER
i2b2.hive.tempCellsList = [
     { code: "PM", ...},
     { code: "ONT", ...},
     { code: "CRC", ...},
     { code: "WORK", ...},
     { code: "PLUGINMGR", ...},
     { code: "Dem1Set",
       forceLoading: true,
       forceConfigMsg: { params: [] },
       roles: [ "DATA_LDS", "DATA_DEID", "DATA_PROT" ],
       forceDir: "cells/plugins/standard"
     },
     { code: "ExportXLS",
       forceLoading: true,
       forceConfigMsg: { params: [] },
       roles: [ "DATA_LDS", "DATA_DEID", "DATA_PROT" ],
```

```
        forceDir: "cells/plugins/community"
      }
   ];
//========================================================================//
```

Each plugin will have a configuration object within the defined **i2b2.hive.tempCellsList** array variable. The meaning of the configuration object's attributes is as follows:
The **code** attribute is the name of the plugin as well as the name of the directory containing the plugin's code and assets.
The **forceLoading** attribute notifies the legacy i2b2 plugin execution environment that the plugin should be loaded upon environment creation. For the plugin to work correctly this must be set to true.
The **forceConfigMsg** attribute contains configuration information that is passed to the plugin. It is an object which *must* contain an array named params.
The **roles** attribute contains the roles which can use the plugin. The rolls are as follows: DATA_OBFSC for Obfuscated, DATA_AGG for Aggregated, DATA_LDS for Limited Data Set, DATA_DEID for De-identified Data, DATA_PROT for Protected.
The **forceDir** attribute points to the parent directory that the plugin's directory exists within.

MODIFYING OF LOGIN SCREEN

Set Address for Legacy Client Link

By default the login screen contains a link for users to access the legacy web client. By default it has no address set. If you are still running the legacy web client then you should enter the URL that it is hosted at. If you are not hosting the legacy web client then remove the link from the template. The HTML template file is located at **/js-i2b2/cells/PM/assets/login.html**
For more information on changing the HTML and/or the look and feel of the new web client refer to the document titled "*Applying Branding Styles to the I2B2 Web Client*".

Other Suggested Readings

- For troubleshooting tips, see i2b2 Usage Community FAQ
- Upgrade Guide for i2b2 Release v1.8
- Applying Branding Styles to the i2b2 Release v1.8+
- Plugin Development for i2b2 Release v1.8+

i2b2 Web Client Repository

https://github.com/hms-dbmi/i2b2v2-webclient

i2b2 Web Proxy Server Repository

https://github.com/hms-dbmi/i2b2-webclient-proxy

Troubleshooting

Troubleshooting Odd Characters in Query Results

If obfuscated users are encountering an unusual "A-hat" character (Â) in their search results, it's likely due to a character encoding issue within your Apache server. To resolve this, you'll need to enable UTF-8 encoding.

Solution:

1. **Locate the `charset.conf` file:**
    - This file is typically located in your Apache configuration directory. Its exact path may vary depending on your server setup. Common locations include:
        - `/etc/apache2/conf/charset.conf`

- `/etc/httpd/conf/charset.conf`

2. **Uncomment the UTF-8 line:**
   - Open the `charset.conf` file using a text editor.
   - Find the line that reads:
     ```
     AddDefaultCharset UTF-8
     ```
   - If this line is commented out (preceded by a `#` symbol), remove the `#` to uncomment it. This will instruct Apache to use UTF-8 as the default character set.
3. **Restart Apache:**
   - After making the change, restart your Apache server to apply the configuration. The specific command to restart Apache will depend on your operating system and server setup. Common commands include:
     - **Linux:** `sudo systemctl restart apache2` or `sudo service apache2 restart`
     - **macOS:** `sudo apachectl restart`

Acknowledgments

Nick Benik, Marc-Danie Nazaire, Marc Ciriello, Anupama Maram, Perminder Singh, Griffin Weber