# Network-specific: SCILHS to 1.19.2

SCILHS sites should only need to upgrade the SHRINE webclient, shrine.war, and shrine-proxy.war. The SHRINE Data Steward and its related installation /upgrade tasks are not used on the SCILHS SHRINE network. We recommend that you upgrade i2b2 to 1.7.05 if at all possible, but it is not an absolute necessity. These instructions are based on the instructions in the general Upgrading SHRINE to 1.22.8 (from 1.20-1.21) article.



## Shut Down SHRINE

Before starting the upgrade process, make sure SHRINE's Tomcat is not running. Leaving it running during this process can cause problems. Simply run the following command:

```
$ shrine_shutdown
```

If the above command is not found, try the following instead:

```
$ /opt/shrine/tomcat/bin/shutdown.sh
```

## Create Backups

Now that SHRINE is stopped, it is a good idea to back up the current versions of the components we will be upgrading. The exact method for making this backups may vary, but these instructions will place the backups in a folder called /opt/shrine/upgrade-backups.

Start by creating a folder to contain these backups:

```
$ mkdir /opt/shrine/upgrade-backups
```

Next, move the current SHRINE webapp to the backup location:

```
$ mv /opt/shrine/tomcat/webapps/shrine /opt/shrine/upgrade-backups/shrine
```

Next, move the current SHRINE proxy to the backup location:

```
$ mv /opt/shrine/tomcat/webapps/shrine-proxy /opt/shrine/upgrade-backups/shrine-proxy
```

Next, move the SHRINE webclient to that same backup location. Later on, we will be restoring i2b2_config_data.js from this backup. If you choose not to make any backups, make sure to at least keep a copy of **i2b2_config_data.js** and **js-i2b2/cells/SHRINE/cell_config_data.js**!

```
$ mv /opt/shrine/tomcat/webapps/shrine-webclient /opt/shrine/upgrade-backups/shrine-webclient
```

## [NEW] Upgrade Tomcat

You will need to upgrade Tomcat to version 7. Most older SHRINE installations defaulted to Tomcat 6, which is incompatible with the new Data Steward application. To ease the upgrade process, the SHRINE installer includes a script to detect the presence of Tomcat 6 and upgrade it to Tomcat 7. It will also back up your existing Tomcat installation, just in case.

Before running the upgrade script, make sure the following environment variables are set correctly:

- SHRINE_HOME
- SHRINE_TOMCAT_HOME
- SHRINE_PORT
- SHRINE_SSL_PORT

- KEYSTORE_FILE
- KEYSTORE_PASSWORD

Next, download the upgrade script from the SHRINE installer and run it.

```
$ wget --no-check-certificate https://open.med.harvard.edu/svn/shrine/releases/1.19.2/code/install/i2b2-1.7/shrine/upgrade_tomcat.sh
$ chmod +x upgrade_tomcat.sh
$ ./upgrade_tomcat.sh
```

The script should cleanly replace an existing Tomcat 6 installation with Tomcat 7, generating a new server.xml (based on tomcat7_server.xml) in the process. If you are already on Tomcat 7 (or newer), the script will exit and do nothing.

If Tomcat fails to start, check /opt/shrine/tomcat/logs/catalina.out. An error like this may appear in catalina.out, especially if you are using OpenJDK:

```
java.lang.ClassNotFoundException: org.apache.catalina.mbeans.ServerLifecycleListener
```

If this appears, comment out the following line in /opt/shrine/tomcat/conf/server.xml:

```
<Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"/>
```

## Deploy New shrine.war

Next, we will retrieve the new SHRINE webapp from the HMS Sonatype Nexus server at http://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/shrine-war/. Navigate to the folder for 1.19.2. From there, download the shrine-war-1.19.2.war file to the webapps directory on the SHRINE server and rename it to shrine.war.

For example:

```
$ cd /opt/shrine/tomcat/webapps
$ wget http://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/shrine-war/1.19.2/shrine-war-1.19.2.war -O shrine.war
```

## Deploy New shrine-proxy.war

The SHRINE proxy can be found on the HMS Sonatype Nexus server at https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/shrine-proxy/. Navigate to the folder for 1.19.2. From there, download the shrine-proxy-1.19.2.war to the webapps directory on the SHRINE server and rename it to shrine-proxy.war.

For example:

```
$ cd /opt/shrine/tomcat/webapps
$ wget https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/shrine-proxy/1.19.2/shrine-proxy-1.19.2.war -O shrine-proxy.war
```

## Deploy New SHRINE Webclient

Unlike shrine.war and steward.war, the SHRINE webclient is retrieved from the releases folder of the HMS Subversion repository at https://open.med.harvard.edu/svn/shrine/releases/. The webclient is found at 1.19.2/code/shrine-webclient. Checkout or export this folder to /opt/shrine/tomcat/webapps.

For example:

```
$ cd /opt/shrine/tomcat/webapps
$ svn export https://open.med.harvard.edu/svn/shrine/releases/1.19.2/code/shrine-webclient/
```

### Restore Webclient Backups

After this, restore the previous **i2b2_config_data.js** and **cell_config_data.js** files from your backup and place them in the new shrine-webclient folder:

```
$ cp /opt/shrine/upgrade-backups/shrine-webclient/i2b2_config_data.js /opt/shrine/tomcat/webapps/shrine-webclient/i2b2_config_data.js
```

```
$ cp /opt/shrine/upgrade-backups/shrine-webclient/js-i2b2/cells/SHRINE/cell_config_data.js /opt/shrine/tomcat/webapps/shrine-webclient/js-i2b2/cells/SHRINE/cell_config_data.js
```

## [NEW] Deploy New steward.war

Much like shrine.war, the SHRINE Data Steward can be found on the HMS Sonatype Nexus server at http://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/steward/. Navigate to the folder for 1.19.2. From there, download the steward-1.19.2.war file to the webapps directory on the SHRINE server and rename it to steward.war.

For example:

```
$ cd /opt/shrine/tomcat/webapps
$ wget --no-check-certificate https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/steward/1.19.2/steward-1.19.2.war -O steward.war
```

# [NEW] Configure the SHRINE Data Steward

## Database - i2b2

The SHRINE Data Steward is typically backed by the i2b2 PM cell used by SHRINE. From the steward application's point of view, all users on the SHRINE project are considered Researchers (except for the Steward and QEP users!). However, there is some additional work that has to be done to the i2b2 user list to accommodate the SHRINE Data Steward.

### QEP User

The Steward application requires set of user credentials that the application will use to submit queries through to SHRINE. It is recommended that this be a dedicated system user separate from any other account. Additionally, it will need to have the parameter "qep" defined (name: qep, value: true, type: text), which can be set in the Manage Users section of the i2b2 Admin Panel. Make sure to add this user to the **SHRINE** project in the i2b2 Admin Panel as well.

In shrine.conf, make sure there is a **shrineSteward** block in the **queryEntryPoint** section, and that the **qepUserName** and **qepPassword** properties match the user with the qep parameter.

### Steward User

In Steward application deployments that involve topic approval, a trusted human user will have to be given permission to review proposed research topics and approve/reject them. To mark a user as such, add the "DataSteward" parameter (name: DataSteward, value: true, type: text) to that user in the Manage Users section of the i2b2 Admin Panel. Make sure to add this user to the **SHRINE** project in the i2b2 Admin Panel as well.

Note that this user is meant solely to monitor and review, and thus cannot create its own query topics! If you are both a steward and a researcher, you will need to create two separate i2b2 users.

#### All Other Users (Researchers)

**NOTE:** The current SHRINE Data Steward webclient transforms usernames to all-lowercase! As such, any i2b2 usernames that use both uppercase and lowercase letters will not be able to use the SHRINE Data Steward properly. Please make sure any users that intend to use SHRINE have a username with all lowercase letters. We apologize for any inconvenience this may cause.

## Database - Steward

The SHRINE Data Steward application uses a separate database to store its logging information, similar to how the core SHRINE application keeps its own query log.

This step should be handled by install-steward-only.sh, but instructions to manually perform it can be found below:

1. Create a database (typically stored in MySQL alongside the existing shrine_query_history database, but this is just a suggestion). The installer by default calls it stewardDB.
2. Grant a shrine/steward-specific user full access to this database. The installer by default uses the same credentials as it does for the SHRINE query history database, but you are welcome to (and encouraged to!) change this.
3. Run the appropriate schema script included with the steward application
    a. For MySQL, this is https://open.med.harvard.edu/svn/shrine/releases/1.19.2/code/steward/src/main/sql/mysql.ddl
    b. For SQL Server, this is https://open.med.harvard.edu/svn/shrine/releases/1.19.2/code/steward/src/main/sql/sqlserver.ddl

## steward.conf - Application config

1. Start with a sample steward.conf from the installer: https://open.med.harvard.edu/svn/shrine/releases/1.19.2/code/install/i2b2-1.7/shrine/skel/steward.conf
    a. If using the JDBC driver (for example, SQL Server users), make sure slickProfileClassName is set to "scala.slick.driver.JdbcDriver$" (mind the lowercase "dbc"!)
2. Substitute the following variables with their appropriate value. (see common.rc and shrine.rc for reference, as well as the listing of configuration variables in the installer)
    a. SHRINE_ADAPTER_I2B2_DOMAIN
    b. SHRINE_STEWARD_DB_NAME
    c. I2B2_PM_IP
    d. KEYSTORE_FILE
    e. KEYSTORE_PASSWORD
    f. KEYSTORE_ALIAS
3. Save this file to /opt/shrine/tomcat/lib/steward.conf

Make sure that the keystore{} block in steward.conf **exactly** matches the keystore{} block in shrine.conf!

An important setting in steward.conf which should be manually set by a site administrator is **createTopicsMode**. By default, this is set to "**Pending**", which means that topic requests submitted in the Data Steward application will have to be manually approved by a user with DataSteward privileges.

If your site has not yet established a dedicated user to review topic submissions, set **createTopicsMode** to "**Approved**". In this mode, topic requests must be submitted, but no manual approval is necessary. All topics submitted are automatically approved, and researchers can immediately begin executing queries for that topic. However, if your site does have a user assigned to act as a Steward, set **createTopicsMode** to "**Pending**".

## steward.xml - Tomcat context definition

1. Start with a sample steward.xml from the installer: https://open.med.harvard.edu/svn/shrine/releases/1.19.2/code/install/i2b2-1.7/shrine/skel/steward.xml
2. Substitute the following variables with their appropriate value. (see common.rc and shrine.rc for reference, as well as the listing of configuration variables in the installer)
    a. SHRINE_STEWARD_MYSQL_USER
    b. SHRINE_STEWARD_MYSQL_PASSWORD
    c. SHRINE_STEWARD_MYSQL_HOST
    d. SHRINE_STEWARD_DB_NAME
3. Save this file to /opt/shrine/tomcat/conf/Catalina/localhost/steward.xml

### SHRINE Webclient

In webapps/shrine-webclient/js-i2b2/cells/SHRINE/**cell_config_data.js**, make sure newTopicURL and readApprovedURL are set to something like the following:

```
    newTopicURL: "https://your.hostname.here:6443/steward/client/index.html#/topics", /* this URL should be accessible from the client */
    readApprovedURL:"https://your.hostname.here:6443/shrine/rest/i2b2/request" /* this URL should be accessible from the SHRINE server */
```

Make the following change in webapps/shrine-webclient/**i2b2_config_data.js**:

```
    isSHRINE: true
```

The above may already be present in i2b2_config_data.js, but commented out. (there may be a /* on the previous line, as well as a */ or // on the "isSHRINE" line) If so, uncomment it. If **isSHRINE** is not already present, add it to the main entry in **lstDomains**.

## [NEW] shrine.conf Changes

### SHRINE Data Steward Integration

In order to make SHRINE aware of the steward, you will need to change the authorizationType value and add the following shrineSteward block to the queryEntryPoint block of shrine.conf:

```
queryEntryPoint {
  ...

  authorizationType = "shrine-steward"

  shrineSteward {
    qepUserName = "qep" // name of user the steward will submit queries as
    qepPassword = "qep-password"
    stewardBaseUrl = "https://your.hostname.here:6443" // typically hostname+port of Tomcat server running steward.war
  }

  ...
}
```

The qepUserName and qepPassword properties should match the set of credentials used for the QEP user defined earlier.

## Start SHRINE

The only thing left to do at this point is start SHRINE back up. Simply do the following:

```
$ shrine_startup
```

If the above command is not found, try the following instead:

```
$ /opt/shrine/tomcat/bin/startup.sh
```

## Verify SHRINE Upgrade

After starting SHRINE up, verify that the upgrade was properly deployed by checking the SHRINE Happy module. The exact address you will need to go to depends on your configuration, but the general format looks like the following:

```
https://your.shrine.host:6443/shrine/rest/happy/version
```

It may take a few seconds for the page to load, but after it does load, verify that the value for <shrineVersion> matches the version that was just deployed. If it is still displaying an old version, repeat steps 1 and 4 to redeploy the shrine.war file and start SHRINE again. Example output from this report for SHRINE 1.19.2 can be seen below:

```
<versionInfo>
  <shrineVersion>1.19.2</shrineVersion>
  <ontologyVersion>UNKNOWN</ontologyVersion>
  <adapterMappingsVersion>Unknown</adapterMappingsVersion>
  <scmRevision>(not available)</scmRevision>
  <scmBranch>UNKNOWN_BRANCH</scmBranch>
  <buildDate>2015-06-30 18:03:57</buildDate>
</versionInfo>
```