Upgrading SHRINE to 1.19.x

In most cases, upgrading an existing instance of SHRINE is a relatively quick process. Exceptions to this rule include older versions of SHRINE that contained substantial changes to configuration files and other portions of the file structure. The instructions here specifically describe an upgrade path from SHRINE 1.18.x to SHRINE 1.19.x. For instructions on upgrading to SHRINE 1.18.x, check the child pages underneath this article.

If upgrading from SHRINE 1.17.x to SHRINE 1.19.x, you will need to make sure to run the database scripts listed in the "Schema Changes" section of the 1.18.x article in addition to the instructions provided in this 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 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
```

Upgrade Tomcat

If you plan to use the Data Steward application, 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 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 in the process. If you are already on Tomcat 7 (or newer), the script will exit and do nothing.

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/. Choose the version that you are upgrading to (for example, 1.19.2) and navigate to that folder. From there, download the appropriate shrine-war-[VERSION].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

Like other SHRINE artifacts, 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
```

[Optional] Deploy New steward.war

Much like shrine.war, steward.war 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 appropriate steward-[VERSION].war file to the webapps directory on the SHRINE server and rename it to steward.war.

For example:

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

After this, restore the previous i2b2_config_data.js and cell_config_data.js file from your backup and place it 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
```

[Optional] Configure the SHRINE Data Steward

Note that a few of these steps can be automatically done by the install-steward-only.sh script available with the SHRINE installer.

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. However, there is some additional work that has to be done to the i2b2 user list to accommodate the SHRINE Data Steward.

NOTE: **install-steward-only.sh** does **NOT** handle this! The creation of i2b2 users/setting of user parameters must be done manually by an i2b2 administrator!

QEP User

The Steward application requires set of user credentials that it will use to submit queries through to SHRINE. It is recommended that this be a dedicated 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.

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 require manual topic approval, a trusted 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.

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

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

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, 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

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. The potential values for **createTopicsMode** are:

- **Pending**: Topic requests must be submitted, and a user with the DataSteward parameter set for them in the i2b2 PM cell must log in and approve them manually before queries can be executed under that topic.
- **Approved**: 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.
- **TopicsIgnoredJustLog**: No topic requests are necessary, the Data Steward application simply intercepts queries for recordkeeping and analysis. If using this mode, ensure the **isSHRINE** property in shrine-webclient/i2b2_config_data.js is either not present or set to **false**

steward.xml - Tomcat context definition

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

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

NOTE: **install-steward-only.sh** does **NOT** handle this step!

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",
    readApprovedURL:"https://your.hostname.here:6443/shrine/rest/i2b2/request"
```

If **createTopicsMode** in steward.conf is set to either "Approved" or "Pending", then 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) If so, uncomment it. If **isSHRINE** is not already present, add it to the main entry in **lstDomains**.

shrine.conf Changes

SHRINE Data Steward Integration

If using the SHRINE Data 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 = "trustme"
    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.

Steward Naming Change (HMS Only)

Users on the HMS SHRINE networks will have to make a change to shrine.conf in order to accommodate a renamed property value.

In the queryEntryPoint block, change authorizationType from "data-steward" to "hms-steward":

```
queryEntryPoint {
  ...

  authorizationType = "hms-steward"

  ...
}
```

It is also worth noting that HMS users will not need to deploy or configure the new SHRINE Data Steward, so they may safely ignore steps involving the SHRINE Data Steward.

Breakdowns

In SHRINE 1.18.0+, it is now required to specify the names of result types corresponding to breakdown queries. These names must match the names of result output types defined in the i2b2 DB of every node on your Shrine network. For example, on a network comprised of nodes backed by i2b2 demo VMs, add this to shrine.conf:

```
shrine {
  ...
  breakdownResultOutputTypes {
    PATIENT_AGE_COUNT_XML {
```

```
      description = "Age patient breakdown"
    }

    PATIENT_RACE_COUNT_XML {
      description = "Race patient breakdown"
    }

    PATIENT_VITALSTATUS_COUNT_XML {
      description = "Vital Status patient breakdown"
    }

    PATIENT_GENDER_COUNT_XML {
      description = "Gender patient breakdown"
    }
  }
  ...
}
```

the format is

```
breakdownResultOutputTypes {
  <breakdown-result-output-type-name 0> {
    description = <string human-readable-description 0>
  }
  ...
    <breakdown-result-output-type-name N> {
    description = <string human-readable-description N>
  }
}
```

Note that shrine.breakdownResultOutputTypes can contain 0 or more child elements.

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 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-server.name.here: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>
```