Upgrading SHRINE to 2.0.0-PR1 (from version 1.25.4)

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 1.25.4 to SHRINE 2.0.0-PR1.

> **Warning**
>
> SHRINE 2.0.0-PR1 is not backwards-compatible with any previous version!

This guide makes the following assumptions of a current 1.25.4 system. Make sure all of these conditions are satisfied before proceeding:

- i2b2 1.7.09c is installed and operational.
- SHRINE 1.25.4 is installed and operational.
- The SHRINE Data Steward has been installed and configured properly.
- The SHRINE Dashboard has been installed and configured properly.

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, especially with unpacking new .war files. Simply run the following command:

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

Backup your current SHRINE configuration

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
```

Make especially sure that the **shrine-webclient/** folder is backed up. 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
```

Make especially sure that the shrine.keystore is backed up. If you lose the private side of a cert you may not be able to recover it.

```
$ cp /opt/shrine/shrine.keystore /opt/shrine/upgrade-backups/shrine.keystore
```

Remove old SHRINE components

SHRINE will use new components, so to avoid conflicts with older files, you can remove the old .war files with this command:

```
$ rm /opt/shrine/tomcat/webapps/*.war
```

Also, you can remove all the existing folders from the /webapps directory:

```
$ rm -rf /opt/shrine/tomcat/webapps/shrine/
$ rm -rf /opt/shrine/tomcat/webapps/shrine-proxy/
$ rm -rf /opt/shrine/tomcat/webapps/shrine-metadata/
$ rm -rf /opt/shrine/tomcat/webapps/shrine-dashboard/
$ rm -rf /opt/shrine/tomcat/webapps/steward/
```

Deploy New .war Files

Deploy New SHRINE Core

Next, we will retrieve the new SHRINE war files from the HMS Sonatype Nexus server at: https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/shrine-war/2.0.0-PR1/. From there, download **shrine-war-2.0.0-PR1.war** to the **webapps/** directory on the SHRINE server and rename it to **shrine.war**.

For example:

```
$ cd /opt/shrine/tomcat/webapps
$ wget https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/shrine-war/2.0.0-PR1/shrine-war-2.0.0-PR1.war -O shrine.war
```

Deploy New SHRINE API Service

Like other SHRINE artifacts, the SHRINE API Service can be found on the HMS Sonatype Nexus server at https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/shrine-api-war/. From there, download **shrine-api-war-2.0.0-PR1.war** to the **webapps/** directory on the SHRINE server and rename it to **shrine-api.war**.

For example:

```
$ cd /opt/shrine/tomcat/webapps
$ wget https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/shrine-api-war/2.0.0-PR1/shrine-api-war-2.0.0-PR1.war -O shrine-api.war
```

Changes to SHRINE's Configuration File - shrine.conf

Since SHRINE 1.25.4, there are a couple of changes to SHRINE's configuration file. If you want to, skip ahead to the sample shrine.conf section and copy the entire contents of the file for use (replacing with your own site specific values).

Part of the new data service allows a node to look up information about itself

Each node can ask the hub about itself and the hub. To ask about itself it needs a shared key with the hub. This will be a unique identifier, something like "HarvardProdNode". Add that to shrine.conf:

```
shrine {
  nodeKey = "UniqueFriendlyNodeName"
}
```

Remove the QEP user from the PM cell, and corresponding section from shrine.conf

Shrine now accesses the DSA's database directly from inside shrine-api.war. It no longer needs a special QEP user. Delete that user from the PM cell, and remove this section from shrine.conf:

```
//shrine-steward config
shrineSteward {
  qepUserName = "qep"
  qepPassword = "changeit"
  stewardBaseUrl = "https://shrine-node:6443"
}
```

Remove includeAggregateResults

Remove the formerly required "includeAggregateResults = false" from shrine.conf .

```
includeAggregateResults = false
```

Configure the webclient in shrine.conf

In SHRINE 2.0.0-PR1, you will need additional parameters to configure the behavior of the webclient. Please add this section in shrine.conf:

```
webclient {
   domain = "i2b2demo"
   name  = "SHRINE"
   siteAdminEmail = "email@example.com"
   usernameLabel = "User Name"
   passwordLabel = "User Password"
   queryFlaggingInstructions = "Enter instructions for flagging queries here"
   flaggingPlaceholderText = "Enter placeholder text for the query flagging text input field"
   flaggingIconInstructions = "Enter text for when user mouses over the flagging information icon in the header
of the Query History here"
 }
```

The webclient has hard-coded default values for the userNameLabel, passwordLabel, and defaultNumberOfOntologyChildren fields if they are not configured; userNameLabel defaults to "SHRINE User", passwordLabel defaults to "SHRINE Password", and defaultNumberOfOntologyChildren defaults to 10000.

Remove Human-Readable node name property from shrine.conf

The nodeKey property is used by SHRINE internally instead, and the node structure (stored at the hub) holds a real human-readable name with a good assortment of punctuation and spaces, therefore, you will not need this line in shrine.conf anymore.

```
humanReadableNodeName = "Harvard Test Node"
```

Redirect webclient, steward, and dashboard to old URL links

The URL to the SHRINE Webclient has changed from https://shrine_url:6443/shrine-webclient to https://shrine_url:6443/shrine-api/shrine-webclient.

The URL to the SHRINE Steward has changed from https://shrine_url:6443/steward to https://shrine_url:6443/shrine-api/shrine-steward.

The URL to the SHRINE Dashboard has changed from https://shrine_url:6443/shrine-dashboard to https://shrine_url:6443/shrine-api/shrine-dashboard.

To redirect the old URL and preserve existing bookmarks and history in browsers, specify a URL rewrite in Tomcat.

1. Add rewrite valve to conf/server.xml inside the Host configuration section

<Host name="localhost" appBase="webapps"
unpackWARs="true" autoDeploy="true">
<Valve className="org.apache.catalina.valves.rewrite.RewriteValve" />

2. Create a file name *rewrite.config* in conf/Catalina/localhost with the following contents

RewriteRule ^/shrine-webclient/* /shrine-api/shrine-webclient
RewriteRule ^/steward/* /shrine-api/shrine-steward
RewriteRule ^/shrine-dashboard/* /shrine-api/shrine-dashboard

No need to specify a slick driver anymore

Now the system can select the slick driver based on the shrineDatabaseType property in shrine.conf.

If the selected slick driver causes problems it is OK to use the old properties - which will override the one chosen via shrineDatabaseType.

Sample SHRINE configuration file (for a downstream node)

Here's a sample shrine.conf for a 2.0.0-PR1 downstream node:

shrine.conf

```
shrineHubBaseUrl = "https://shrine_hub_url:6443"  //The shrine hub's URL as observed from this tomcat server
i2b2BaseUrl = "http://localhost:9090"             //The local i2b2's URL as observed from this tomcat server
i2b2Domain = "i2b2demo"                           //recommended to change this to a unique domain
i2b2ShrineProjectName = "SHRINE"

shrine {

  nodeKey = "unique-node-name"         //node key to get information from the hub about this node

  shrineDatabaseType = "mysql"         // default
                                                        //can be oracle, mysql, sqlserver

  messagequeue {
    blockingq {
      serverUrl = ${shrineHubBaseUrl}/shrine-api/mom  //point this to the network hub
    }
  }
```

```
webclient {
 domain = ${i2b2Domain}
 name  = ${i2b2ShrineProjectName}
 siteAdminEmail = "shrine-admin@example.com"
}

pmEndpoint {
  url = ${i2b2BaseUrl}/i2b2/services/PMService/getServices
}
ontEndpoint {
  url = ${i2b2BaseUrl}/i2b2/services/OntologyService/
}

hiveCredentials {
  domain = ${i2b2Domain}
  username = "demo"
  password = "demouser"
  crcProjectId = "Demo"
  ontProjectId = ${i2b2ShrineProjectName}
}

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"
  }
} //end breakdown section

hub {
  client {
    serverUrl = ${shrineHubBaseUrl}
  }
}

queryEntryPoint {
  broadcasterServiceEndpoint {
    url = ${shrineHubBaseUrl}/shrine/rest/broadcaster/broadcast
  }
}

adapter {
  crcEndpoint {
    url = ${i2b2BaseUrl}/i2b2/services/QueryToolService/
  }

  adapterMappingsFileName = "AdapterMappings.csv"
} //end adapter section

keystore {
  file = "/opt/shrine/shrine.keystore"
  password = "password"
  privateKeyAlias = "your_private_key_alias"
  keyStoreType = "JKS"
  caCertAliases = ["shrine-hub-ca"]
} //end keystore section

steward {
  createTopicsMode = Approved

  emailDataSteward {
    //provide the email address of the shrine node system admin, to handle bounces and invalid addresses
    from = "shrine-admin@example.com"
    //provide the email address of the shrine node system admin, to handle bounces and invalid addresses
```

```
      to = "shrine-steward@example.com"
      //provide the externally-reachable URL for the data steward
      externalStewardBaseUrl = ${shrineHubBaseUrl}/shrine-api/shrine-steward
    }


        database {
                dataSourceFrom = "JNDI"
              jndiDataSourceName = "java:comp/env/jdbc/stewardDB"
              slickProfileClassName = "slick.jdbc.MySQLProfile$"
                                //slick.jdbc.H2Profile$
                                //slick.jdbc.PostgresProfile$
                                //slick.jdbc.SQLServerProfile$
                                //slick.jdbc.JdbcProfile$
                                //slick.jdbc.OracleProfile$
    } //end steward database section
  } //end steward section
} //end shrine
```

Changes to SHRINE databases

In SHRINE 2.0.0-PR1, there are a couple of database changes that you will need to perform to accommodate the new tables and structures needed for SHRINE to work properly.

Remove the PRIVILEGED_USER table and associated constraints and sequences from your database

```
use shrine_query_history;
DROP TABLE PRIVILEGED_USER;
```

Drop the column of actual patient counts from the adapter's COUNT_RESULT table

Shrine no longer needs to store the actual (and mildly sensitive) actual count of patients from the CRC in its COUNT_RESULT table. In mysql remove it from the shrine_query_history database with

```
use shrine_query_history;
ALTER TABLE COUNT_RESULT DROP COLUMN ORIGINAL_COUNT;
```

Drop the column of actual patient counts from the adapter's BREAKDOWN_RESULT table

Shrine no longer needs to store the actual (and mildly sensitive) actual count of patients from the CRC in its BREAKDOWN_RESULT table. In mysql remove it from the shrine_query_history database with

```
use shrine_query_history;
ALTER TABLE BREAKDOWN_RESULT DROP COLUMN ORIGINAL_VALUE;
```

Add a status column for queries at the QEP

Add a status column to qepAuditDB's previousQueries table to support updates in query status with :

MySQL:

```
use qepAuditDB;
ALTER TABLE previousQueries ADD COLUMN status VARCHAR(255) NOT NULL DEFAULT 'Before V26';
```

MS SQL:

```
use qepAuditDB;
ALTER TABLE previousQueries ADD COLUMN 'status' VARCHAR(MAX) NOT NULL DEFAULT 'Before V26';
```

ORACLE:

```
use qepAuditDB;
ALTER TABLE "previousQueries" ADD COLUMN "status" VARCHAR2(256) DEFAULT 'Before V26' NOT NULL;
```

Added table QUERY_PROBLEM_DIGESTS

MySQL:

```
use qepAuditDB;
create table QUERY_PROBLEM_DIGESTS (NETWORKQUERYID BIGINT NOT NULL, CODEC TEXT NOT NULL, STAMP TEXT NOT NULL,
SUMMARY TEXT NOT NULL, DESCRIPTION TEXT NOT NULL, DETAILS TEXT NOT NULL, CHANGEDATE BIGINT NOT NULL);
create index queryProblemsNetworkIdIndex on QUERY_PROBLEM_DIGESTS(NETWORKQUERYID);
```

MS SQL:

```
use qepAuditDB;
create table "QUERY_PROBLEM_DIGESTS" ("NETWORKQUERYID" BIGINT NOT NULL,"CODEC" VARCHAR(MAX) NOT NULL,"STAMP"
VARCHAR(MAX) NOT NULL,"SUMMARY" VARCHAR(MAX) NOT NULL,"DESCRIPTION" VARCHAR(MAX) NOT NULL,"DETAILS" VARCHAR
(MAX) NOT NULL,"CHANGEDATE" BIGINT NOT NULL);
create index "queryProblemsNetworkIdIndex" on "QUERY_PROBLEM_DIGESTS" ("NETWORKQUERYID");
```

ORACLE:

```
use qepAuditDB;
create table "QUERY_PROBLEM_DIGESTS" ("NETWORKQUERYID" NUMBER(19) NOT NULL,"CODEC" VARCHAR(256) NOT NULL,"
STAMP" VARCHAR(256) NOT NULL,"SUMMARY" CLOB NOT NULL,"DESCRIPTION" CLOB NOT NULL,"DETAILS" CLOB NOT NULL,"
CHANGEDATE" NUMBER(19) NOT NULL);
create index "queryProblemsNetworkIdIndex" on "QUERY_PROBLEM_DIGESTS" ("NETWORKQUERYID");
```

Added table RESULTS_OBSERVED

MySQL:

```
use qepAuditDB;
create table RESULTS_OBSERVED (NETWORKQUERYID BIGINT NOT NULL, CHECKSUM BIGINT NOT NULL, OBSERVEDTIME BIGINT
NOT NULL);
create index resultsObservedQueryIdIndex on RESULTS_OBSERVED(NETWORKQUERYID);
create index resultsObservedChecksumIndex on RESULTS_OBSERVED(CHECKSUM);
```

MS SQL:

```
use qepAuditDB;
create table `RESULTS_OBSERVED` (`NETWORKQUERYID` BIGINT NOT NULL,`CHECKSUM` BIGINT NOT NULL,`OBSERVEDTIME`
BIGINT NOT NULL);
create index "resultsObservedQueryIdIndex" on "RESULTS_OBSERVED" ("NETWORKQUERYID");
create index "resultsObservedChecksumIndex" on "RESULTS_OBSERVED" ("CHECKSUM");
```

ORACLE:

```
use qepAuditDB;
create table "RESULTS_OBSERVED" ("NETWORKQUERYID" NUMBER(19) NOT NULL,"CHECKSUM" NUMBER(19) NOT NULL,"
OBSERVEDTIME" NUMBER(19) NOT NULL);
create index "resultsObservedQueryIdIndex" on "RESULTS_OBSERVED" ("NETWORKQUERYID");
create index "resultsObservedChecksumIndex" on "RESULTS_OBSERVED" ("CHECKSUM");
```

Start SHRINE

> **PLEASE NOTE**
>
> At this time, please reach out to the network hub administrator so that they can add your individual node to the network with the "nodeKey" that was identified in shrine.conf. Once added to the hub, your node should then start sending and receiving queries.

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

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

Verify SHRINE Upgrade

After starting SHRINE up, verify that the upgrade was properly deployed by checking the SHRINE Dashboard for the version number. 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-dashboard
```

This is a sample screenshot of the upgraded 2.0.0-PR1 dashboard:

Version Info

This site is running SHRINE 2.0.0-PR1 built on **2019-06-26 17:44:39**

This site is currently using **SHRINE** ontology version **UNKNOWN**
Based on concept term: **\\SHRINE\SHRINE\ONTOLOGYVERSION**

This site is currently using **AdapterMappings.csv** for mappings, last edited on **2017-05-17 20:02:01**

System Health

Component	System Health
Keystore:	OK
QEP:	OK
Adapter:	OK