Upgrading SHRINE to 1.18.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 1.17.x to SHRINE 1.18.x.

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
```

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, 1.18.2) 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/1.18.2/shrine-war-1.18.2.war -O shrine.war
```

Deploy New SHRINE Webclient

Unlike shrine.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/1.18.2/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
```

Schema Changes

SHRINE 1.18 adds columns to the SHRINE_QUERY table, and also creates the HUB_QUERY and HUB_QUERY_RESULT tables. Users upgrading from a version before 1.18.0 to 1.18.0 or newer will have to run a few additional database migration scripts.

To upgrade the SHRINE_QUERY table, retrieve the adapter-migrate-schema-to-1.18.sql script and run it:

```
$ wget https://open.med.harvard.edu/svn/shrine/releases/1.18.2/code/adapter/src/main/resources/adapter-migrate-
schema-to-1.18.sql
$ mysql -u $SHRINE_MYSQL_USER -p$SHRINE_MYSQL_PASSWORD -D shrine_query_history < adapter-migrate-schema-to-1.18.
sql
```

To create the HUB_QUERY and HUB_QUERY_RESULT tables, download the hub.sql script and run it:

```
$ wget https://open.med.harvard.edu/svn/shrine/releases/1.18.2/code/broadcaster-aggregator/src/main/resources/hub.
sql
$ mysql -u $SHRINE_MYSQL_USER -p$SHRINE_MYSQL_PASSWORD -D shrine_query_history < hub.sql
```

Shrine.conf Changes

In Shrine 1.18.0+, it is now required to specify the names of result types corresponding to breakdown queries. There 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.

breakdowns.conf

Since this breakdown result output type information will be shared by many nodes on a network, it's also possible to define it in its own file, breakdowns.conf, which can be shared among sites without revealing URLs or credentials. That file should contain only the enclosing shrine { ... } block and the breakdownResultOutputTypes {... } block, like

```
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"
    }
  }
}
```

If breakdowns.conf is present and no shrine.breakdownResultOutputTypes block is defined in shrine.conf, values from breakdowns.conf will be used.

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 1-3 to redeploy the shrine.war file and start SHRINE again. Example output from this report for SHRINE 1.18.2 can be seen below:

```
<versionInfo>
  <shrineVersion>1.18.2</shrineVersion>
  <ontologyVersion>UNKNOWN</ontologyVersion>
  <adapterMappingsVersion>Unknown</adapterMappingsVersion>
  <scmRevision>(not available)</scmRevision>
  <scmBranch>UNKNOWN_BRANCH</scmBranch>
  <buildDate>2015-03-11 09:38:29</buildDate>
</versionInfo>
```