

Upgrading SHRINE to 1.22.8 (from 1.20-1.21)

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 1.20 or 1.21 to SHRINE 1.22.8. For instructions on upgrading to SHRINE 1.20, check the child pages underneath the "Archived Upgrade Instructions" section.

- WARNING: Networks will not be able to run a mix of 1.21 and 1.22.8. Nodes running SHRINE 1.22.8 will not trust messages sent by a node running SHRINE 1.21 or earlier, or vice versa. SHRINE 1.22.8 signs and verifies using the Cryptographic Message Syntax (CMS) standard instead of custom code.

This guide makes the following assumptions of a current 1.20 system. Make sure all of these conditions are satisfied before proceeding:

- i2b2 1.7.08b or newer is installed and operational.
- SHRINE 1.20 or 1.21 is installed and operational.
- The SHRINE Data Steward or later has been installed and configured properly.

- 1 [Before You Start](#)
 - 1.1 [Know Your Network-Wide Configurations for New Features](#)
 - 1.2 [Know Your Local Configurations for New Features](#)
- 2 [Shut Down SHRINE](#)
- 3 [Create Backups](#)
- 4 [Upgrade to JDK 8 and Tomcat 8](#)
 - 4.1 [Remove the environment variable export from Tomcat](#)
- 5 [Deploy New .war Files](#)
 - 5.1 [Deploy New shrine.war](#)
 - 5.2 [Deploy New shrine-proxy.war](#)
 - 5.3 [Deploy New SHRINE Webclient](#)
 - 5.4 [Restore Webclient Backups](#)
 - 5.5 [Deploy New SHRINE Dashboard](#)
 - 5.6 [Deploy New SHRINE Node Metadata Service](#)
- 6 [Keystore Cleanup](#)
- 7 [Database Changes](#)
- 8 [Configure the SHRINE Data Steward](#)
 - 8.1 [Database - i2b2](#)
 - 8.2 [QEP User](#)
 - 8.3 [Steward User](#)
- 9 [Changes to shrine.conf](#)
 - 9.1 [Consolidation of .conf files](#)
 - 9.2 [Configuration Changes](#)
 - 9.2.1 [Missing Default Properties](#)
 - 9.2.2 [Node Meta Data Service](#)
 - 9.2.3 [Configurable Obfuscation](#)
 - 9.2.4 [Turn Off Adapter Lockout](#)
 - 9.2.5 [Bot Defense](#)
 - 9.2.6 [Send Email](#)
 - 9.2.7 [Send Email Audit Requests to the Local Data Steward](#)
 - 9.2.8 [For Users of Oracle or MS SQL Server](#)
 - 9.3 [Tomcat context.xml](#)
- 10 [Start SHRINE](#)
- 11 [Verify SHRINE Upgrade](#)

Before You Start

SHRINE 1.22.8 has new features, some of which require consistent configuration across a network.

Know Your Network-Wide Configurations for New Features

- Which method of re-identification protection does your network use? (required configuration, default provided): In 1.22, we obviate the need for user lockout with new features: greater obfuscation of results, a more interactive Data Steward Application (DSA), and email audit requests from the DSA to the data steward. Lockout remains on by default. (Lockout will be off by default in SHRINE 1.23, and removed in SHRINE 1.24.) Networks should not need both lockout (method employed in 1.21 and prior) and the new re-identification protections introduced in 1.22. We recommend using the new features and turning lockout off if your network governance allows.
- Network contact information (optional configuration): Determine what fields every node should provide via the SHRINE static data service. We created this general-purpose feature to provide contact information for the right person that remote researchers should contact for more information about query results. Networks should agree on key-value pairs to supply for this contact information and any other required information.
- Bot defense values (default values provided, optionally configurable): Determine what values to use for Bot Defense for your network, if other than default.

Know Your Local Configurations for New Features

- Who should get emails generated by SHRINE's Data Steward Application? (optional but highly recommended): Find appropriate email addresses for the system administrators and data stewards.
- How will email from the DSA be sent? (optional but highly recommended): The default configuration assumes localhost is running postfix on port 25.
- If your network has chosen fields for the SHRINE static data service, what are your local values? (optional, local configuration; see "Network contact information" above)

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
```

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
```

Make especially sure that the **shrine-webclient/** folder is backed up. Later on, we will be restoring important webclient configuration files 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
```

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
```

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

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

Make sure to also back up the other existing SHRINE components (shrine-proxy and steward), just in case:

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

Next, you want to backup both the shrine.xml and steward.xml file to the backup folder:

```
$ cp /opt/shrine/tomcat/conf/Catalina/localhost/shrine.xml /opt/shrine/upgrade-backups/shrine.xml
$ cp /opt/shrine/tomcat/conf/Catalina/localhost/steward.xml /opt/shrine/upgrade-backups/steward.xml
```

Finally remove the old .war files with this command:

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

Upgrade to JDK 8 and Tomcat 8

SHRINE 1.22.8 was tested only with JDK 8 and Tomcat 8. Future versions of SHRINE will require JDK 8 and Tomcat 8.

Remove the environment variable export from Tomcat

With JDK 8, tomcat no longer needs a MaxPermSize set to run SHRINE. You may remove the part of **setenv.sh** that set:

setenv.sh contents

```
export CATALINA_OPTS="-Dakka.daemonic=on"
```

or the part of **setenv.bat** that set:

setenv.bat contents

```
set CATALINA_OPTS=-Dakka.daemonic=on
```

Deploy New .war Files

Deploy New shrine.war

Next, we will retrieve the new SHRINE webapp from the HMS Sonatype Nexus server at: <https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/shrine-war/1.22.8/>. From there, download **shrine-war-1.22.8.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/1.22.8/shrine-war-1.22.8.war -O shrine.war
```

Deploy New steward.war

Much like shrine.war, the SHRINE Data Steward can be found on the HMS Sonatype Nexus server at <https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/steward/1.22.8/>. From there, download **steward-1.22.8.war** 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/1.22.8/steward-1.22.8.war -O steward.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/1.22.8/>. From there, download **shrine-proxy-1.22.8.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/1.22.8/shrine-proxy-1.22.8.war -O shrine-proxy.war
```

Deploy New SHRINE Webclient

The SHRINE webclient can be found on the HMS Sonatype Nexus server at <https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/shrine-webclient/1.22.8/>. From there, download **shrine-webclient-1.22.8-dist.zip** file to the **webapps/** directory on the SHRINE server and rename it to **shrine-webclient.zip**. Then, unzip the shrine-webclient.zip file.

For example:

```
$ cd /opt/shrine/tomcat/webapps
$ wget https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/shrine-webclient/1.22.8/shrine-webclient-1.22.8-dist.zip -O shrine-webclient.zip
$ unzip shrine-webclient.zip
```

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
```

Deploy New SHRINE Dashboard

Like other SHRINE artifacts, the SHRINE Dashboard can be found on the HMS Sonatype Nexus server at <https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/dashboard-war/1.22.8/>. From there, download **dashboard-war-1.22.8.war** to the **webapps/** directory on the SHRINE server and rename it to **shrine-dashboard.war**.

For example:

```
$ cd /opt/shrine/tomcat/webapps
$ wget https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/dashboard-war/1.22.8/dashboard-war-1.22.8.war -O shrine-dashboard.war
```

Deploy New SHRINE Node Metadata Service

Like other SHRINE artifacts, the SHRINE Node Metadata Service can be found on the HMS Sonatype Nexus server at <https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/meta-war/1.22.8/>. From there, download **meta-war-1.22.8.war** to the **webapps/** directory on the SHRINE server and rename it to **shrine-meta.war**.

For example:

```
$ cd /opt/shrine/tomcat/webapps
$ wget https://repo.open.med.harvard.edu/nexus/content/groups/public/net/shrine/meta-war/1.22.8/meta-war-1.22.8.war -O shrine-meta.war
```

Keystore Cleanup

For SHRINE 1.22.8, we strongly recommend that the network configuration is a "Hub-and-spoke". For hub-and-spoke systems, we recommend four certificates in SHRINE's shrine.keystore on nodes:

- The node's signing cert so that its QEP can sign queries.
- The hub's public CA cert so that its adapter can verify other nodes' signatures.
- The hub's public https cert so that the node will trust the hub as a server.
- A cert for this node to use to serve https, referenced in the server.xml file. (Use a cert signed by a public cert authority to avoid a warning in users' browsers.)

See what certs are there and remove any extras. SHRINE will verify that the signing cert is signed by the hub's public cert. Also, check that you are not relying on an expired cert. See [Generate a Certificate Signing Request](#).

Be sure the original keystore is backed up!

List and delete a cert

```
$ keytool -list -keystore /opt/shrine/shrine.keystore
$ keytool -delete -noprompt -alias ${cert.alias} -keystore ${keystore.file}
```

You may need to set or update the keyStore keyAlias in server.xml to use your https cert.

```
...
<Connector port="6443" protocol="org.apache.coyote.http11.Http11NioProtocol"
    maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreFile="/opt/shrine/shrine.keystore"
    keystorePass="changeit" keyAlias="your.https.server.cert.alias" />
...
```

Database Changes

These instructions all use mysql syntax. Versions of .ddl files exist for Oracle and MSSQLServer within SHRINE's [source code](#).

SHRINE 1.22.8 adds a table of problems to each SHRINE node, to help diagnose problems observed on that node.

- Add a table named "problems" to the shrine_query_history database

```
/* Working tables in shrine_query_history */
create table `problems` (`id` BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY, `codec` TEXT NOT NULL, `stampText` TEXT NOT NULL, `summary` TEXT NOT NULL, `description` TEXT NOT NULL, `detailsXml` TEXT NOT NULL, `epoch` BIGINT NOT NULL);
create index `idx_epoch` on `problems` (`epoch`);
```

- Expand the limit on the number of characters in a flag message to the database's limit for unbound text in the shrine_query_history database

```
alter table SHRINE_QUERY change flag_message flag_message text;
```

- Change the PRIVILEGED_USER table's threshold column to accept nulls (and use the default threshold after resetting for lockout) in the shrine_query_history database

```
alter table PRIVILEGED_USER change threshold threshold int null;
```

- Add a userAudit table to the DSA's database to store when the data steward has been asked to audit a researcher, to stewardDB .

```
create table `userAudit` (`researcher` VARCHAR(254) NOT NULL, `queryCount` INTEGER NOT NULL, `changeDate` BIGINT NOT NULL);
```

- SHRINE 1.21 added a cache of previous query results to the QEP's qepAuditDB by adding the following tables:

```
create table `previousQueries` (`networkId` BIGINT NOT NULL, `userName` TEXT NOT NULL, `domain` TEXT NOT NULL, `queryName` TEXT NOT NULL, `expression` TEXT, `dateCreated` BIGINT NOT NULL, `deleted` BOOLEAN NOT NULL, `queryXml` TEXT NOT NULL, `changeDate` BIGINT NOT NULL);
create table `queryFlags` (`networkId` BIGINT NOT NULL, `flagged` BOOLEAN NOT NULL, `flagMessage` TEXT NOT NULL, `changeDate` BIGINT NOT NULL);
create table `queryResults` (`resultId` BIGINT NOT NULL, `networkQueryId` BIGINT NOT NULL, `instanceId` BIGINT NOT NULL, `adapterNode` TEXT NOT NULL, `resultType` TEXT, `size` BIGINT NOT NULL, `startDate` BIGINT, `endDate` BIGINT, `status` TEXT NOT NULL, `statusMessage` TEXT, `changeDate` BIGINT NOT NULL);
create table `queryBreakdownResults` (`networkQueryId` BIGINT NOT NULL, `adapterNode` TEXT NOT NULL, `resultId` BIGINT NOT NULL, `resultType` TEXT NOT NULL, `dataKey` TEXT NOT NULL, `value` BIGINT NOT NULL, `changeDate` BIGINT NOT NULL);
create table `queryResultProblemDigests` (`networkQueryId` BIGINT NOT NULL, `adapterNode` TEXT 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);
```

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. 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 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.

Changes to shrine.conf

Consolidation of .conf files

The dashboard.conf, steward.conf, and shrine.conf files have been consolidated into a single shrine.conf file. (The three files had several sections that were duplicated and had to be identical. This should make managing SHRINE's configuration simpler.) All services expect to find all the configuration values they need inside shrine.conf. steward.conf and dashboard.conf will be ignored.

Combine the shrine.conf, steward.conf, and dashboard.conf files into shrine.conf.

Configuration Changes

See the canonical heavily annotated shrine.conf file in [source code control](#).

Missing Default Properties

Add these section within the shrine block to properly log problems

```
problem {
  problemHandler = "net.shrine.problem.LogAndDatabaseProblemHandler$"
}
dashboard {
  gruntWatch = false //false for production, true for mvn tomcat7:run . Allows the client javascript and html
files to be loaded via gruntWatch .
  happyBaseUrl = "https://localhost:6443/shrine/rest/happy"
  statusBaseUrl = "https://localhost:6443/shrine/rest/internalstatus"

  database {
    dataSourceFrom = "JNDI" //Can be JNDI or testDataSource . Use testDataSource for tests, JNDI everywhere
  }
else
  jndiDataSourceName = "java:comp/env/jdbc/problemDB" //or leave out for tests
  slickProfileClassName = "slick.driver.MySQLDriver$" // Can be
  //      slick.driver.H2Driver$
  //      slick.driver.MySQLDriver$
  //      slick.driver.PostgresDriver$
  //      slick.driver.SQLServerDriver$
  //      slick.driver.JdbcDriver$
  //      freeslick.OracleProfile$
  //      freeslick.MSSQLServerProfile$
  //
  //      (Yes, with the $ on the end)
```

```

        createTablesOnStart = false //for testing with H2 in memory, when not running unit tests. Set to false
normally
    }
}

```

If you are using hub-and-spoke SHRINE architecture add this setting to shrine.queryEntryPoint

```

trustModelIsHub = true

attachSigningCert = true

```

Node Meta Data Service

SHRINE networks now include a JSON api at every node that reports about the contents of the metaData section. What belongs in the metaData section is completely configurable, and we recommend that it be used as a means for serving relevant contact information. Bear in mind that this information is publicly available. For more information on interacting with the Node Data Service, please see the [wiki page](#).

```

metaData {
  siteAdminsContactInfo = ["admin1@example.com", "admin2@example.com"]
  dataStewardContactInfo = "data.steward@example.com"
}

```

We strongly encourage networks to agree on key-value pairs for contacting remote SHRINE node system admins, data stewards, and data admins. This new service can share that information across the SHRINE network.

Configurable Obfuscation

Obfuscation parameters are now configurable. Results are rounded to the nearest 5. The default values force a nefarious researcher to run about 30 queries to identify an individual patient, and an additional 30 queries per fact they wish to verify. If you change these values, be sure to change the javascript property that controls the "+- 10" clamp value in SHRINE's web client to match. See [Configuring SHRINE Webclient Obfuscation](#).

```

shrine {
  ...
  adapter {
    ...
    obfuscation {
      // binSize = 5 //by default. Round to the nearest binSize. Use 1 for no effect (to match SHRINE 1.21 and
earlier).
      // sigma = 6.5 //by default. Noise to inject. Use 0 for no effect. (Use 1.33 to match SHRINE 1.21 and
earlier).
      // clamp = 10 //by default. Maximum ammount of noise to inject. (Use 3 to match SHRINE 1.21 and earlier).
    }
  }
  ...
}

```

Turn Off Adapter Lockout

With more obfuscated results monitored by a data steward, networks can turn off adapter lockout. This feature is on by default in 1.22. In 1.23 adapter lockout will be off by default. In 1.24 the code and database column supporting adapter lockout will be removed.

```

shrine {
  ...
  adapter {
    ...
    adapterLockoutAttemptsThreshold = 0 // adapterLockoutAttemptsThreshold = 10 by default // Number of
allowed queries with the same actual result that can exist before a researcher is locked out of the adapter.
Set to '0' to never lock out. In 1.23 the default will change to 0. In 1.24 the lockout code and this config
value will be removed
  }
  ...
}

```

Bot Defense

SHRINE includes a simple mechanism to mitigate the damage that a rogue bot can do if it somehow manages to bypass SHRINE's network security. By default no researcher can run more than 10 queries in one minute, and no more than 200 queries in 10 hours. If you have legitimate bots that operate faster, either turn off this feature or set the counts and time scales to fit those bots' known capability.

```
shrine {
...
  adapter {
...
    //      botDefense {
    //        countsAndMilliseconds = [ //to turn off, use an empty json list
    //          {count = 10, milliseconds = 60000}, //allow up to 10 queries in one minute by default
    //          {count = 200, milliseconds = 36000000} //allow up to 200 queries in 10 hours by default
    //        ]
    //      }
...
}
```

Send Email

We have tested SHRINE's ability to send email via both a local postfix installation and AWS SES. The default configuration should support postfix. You will have to modify the configuration to use AWS SES or other email services. However, SHRINE should be able to use any service that works with a javamail configuration.

```
shrine {
...
  email {
    //add javax mail properties from https://www.tutorialspoint.com/javamail_api/javamail_api_smtp_servers.htm
    here
    //      javaxmail {
    //        mail {
    //          smtp {
    //            //for postfix on localhost
    //            host = localhost
    //            port = 25

    //            //for AWS SES - See http://docs.aws.amazon.com/ses/latest/DeveloperGuide/send-using-smtp-java.html
    //            host = email-smtp.us-east-1.amazonaws.com
    //            port = 25
    //            transport.protocol = smtps
    //            auth = true
    //            starttls.enable = true
    //            starttls.required = true
    //          }
    //        }
    //      }
    //Must be set for AWS SES. See http://docs.aws.amazon.com/ses/latest/DeveloperGuide/send-using-smtp-java.html
    html
    //      authenticator {
    //        username = yourUsername
    //        password = yourPassword
    //      }
  }
...
}
```

Send Email Audit Requests to the Local Data Steward

You must provide email addresses for the "from", "to", and "stewardBaseUrl" fields.

The default values send an audit request to the data steward at 6 AM if any researcher has run more than 30 queries since his last audit, or any researcher has run more than one query in the last 30 days since his last audit. Note that when this system first becomes active the data steward will very likely receive an audit request for queries run in earlier versions of SHRINE.

```
shrine {
...
  steward {
...
    emailDataSteward {
    //      sendAuditEmails = true //false to turn off the whole works of emailing the data steward
    }
  }
}
```



```
//      interval = "1 day" //Audit researchers daily
//      timeAfterMidnight = "6 hours" //Audit researchers at 6 am. If the interval is less than 1 day then this
delay is ignored.
//      maxQueryCountBetweenAudits = 30 //If a researcher runs more than this many queries since the last audit
audit her
//      minTimeBetweenAudits = "30 days" //If a researcher runs at least one query, audit those queries if this
much time has passed

//You must provide the email address of the shrine node system admin, to handle bounces and invalid
addresses
//from = "shrine-admin@example.com"
//You must provide the email address of the data steward
//to = "shrine-steward@example.com"

//      subject = "Audit SHRINE researchers"
//The baseUrl for the data steward to be substituted in to email text. Must be supplied if it is used in
the email text.
//stewardBaseUrl = "https://example.com:8443/steward/"

//Text to use for the email audit.
// AUDIT_LINES will be replaced by a researcherLine for each researcher to audit.
// STEWARD_BASE_URL will be replaced by the value in stewardBaseUrl if available.
//      emailBody = ""Please audit the following users at STEWARD_BASE_URL at your earliest convinience:
//
//AUDIT_LINES"" //note that this can be a multiline message

//Text to use per researcher to audit.
//FULLNAME, USERNAME, COUNT and LAST_AUDIT_DATE will be replaced with appropriate text.
//      researcherLine = "FULLNAME (USERNAME) has run COUNT queries since LAST_AUDIT_DATE."
}
...

```

For Users of Oracle or MS SQL Server

If you experience issues in getting the SHRINE Data Steward (or the new audit logging functionality) working with Oracle or SQL Server, please try the instructions in this article written by a SHRINE developer: [Using SHRINE Data Steward with Oracle or SQL Server](#)

This article contains links to alternative database drivers, as well as links to alternative database scripts for creating tables for the SHRINE Data Steward. It also includes information on the necessary changes to make to **shrine.xml**, **steward.xml**, and **shrine.conf**.

shrine.conf includes several instances of slickProfileClassName. By default, these are configured to use MySQL. Users of Oracle or SQL Server for SHRINE must add them. See the below for an example:

```
shrine {
[...]
  queryEntryPoint {
    audit {
      database {
        slickProfileClassName="freeslick.driverNameHere$"
      }
    }
    [...]
  }
[...]
```

```
  adapter {
    audit {
      database {
        slickProfileClassName="freeslick.driverNameHere$"
      }
    }
    [...]
  }
  dashboard {
    database {
      slickProfileClassName="freeslick.driverNameHere$"
    }
    [...]
  }
}
```

```

steward {
  database {
    slickProfileClassName="freeslick.driverNameHere$"
  }
  [...]
}
[...]
}

```

Replace "freeslick.driverNameHere\$" with the name of the driver you are using ("freeslick.OracleProfile\$" for Oracle, and "freeslick.MSSQLServerProfile\$" for SQL Server).

Tomcat context.xml

To support shrine's improved error message feature, add a context.xml file at /opt/shrine/tomcat/conf/context.xml with contents appropriate for your database:

context.xml

```

<?xml version='1.0' encoding='utf-8'?>
<!-- The contents of this file will be loaded for each web application -->

<Context>
  <WatchedResource>WEB-INF/web.xml</WatchedResource>

  <Resource name="jdbc/problemDB" auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000"
    username="shrine" password="demouser" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/shrine_query_history"
    testOnBorrow="true" validationQuery="SELECT 1"/>

  <Resource name="jdbc/shrineDB" auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000"
    username="shrine" password="demouser" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/shrine_query_history"
    testOnBorrow="true" validationQuery="SELECT 1"/>

  <Resource name="jdbc/adaptorAuditDB" auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000"
    username="shrine" password="demouser" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/adaptorAuditDB"
    testOnBorrow="true" validationQuery="SELECT 1"/>

  <Resource name="jdbc/qpAuditDB" auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000"
    username="shrine" password="demouser" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/qpAuditDB"
    testOnBorrow="true" validationQuery="SELECT 1"/>

  <Resource name="jdbc/stewardDB" auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000"
    username="shrine" password="demouser" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/stewardDB"
    testOnBorrow="true" validationQuery="SELECT 1"/>

</Context>

```

Start SHRINE

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. 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
```

It may take a few seconds for the page to load, but after it does load, log in with your SHRINE credentials (any user will do, regardless of role) and verify that the value for "SHRINE Version" is 1.22.8. If it is still displaying an old version, repeat the instructions in the "**Deploy New shrine.war**" section, restart SHRINE, and try again.