Database (SQL) Realm

{scrollbar}

In this section we will focus on the use a database for verifying and retrieving user names and passwords.

For this example we created a new database called **SecurityDatabase** using the built-in Derby database. The following steps summarize the procedure performed to create the database and tables, load some sample data and create the connection pool. Detailed instructions on how to define database connection pools are described in the Configuring database pools section.

Create database and load sample data

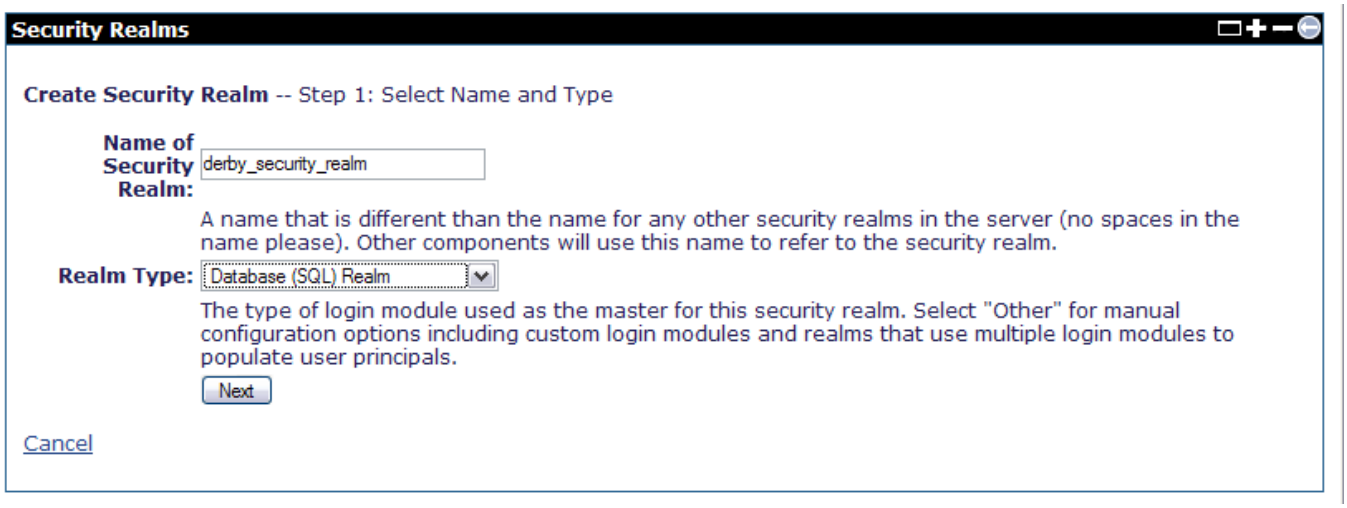
- In the **Console Navigation** menu on the left click on **DB Manager**.
- Enter **SecurityDatabase** in the **Create DB:** field and click **Create**.
- Select the **SecurityDatabase** database from the **Use DB:** pull-down menu, enter the following commands and click **Run SQL**. SQLsolid create table users(username varchar(15), password varchar(15)); create table groups(username varchar(15), groupname varchar(15)); insert into users values('userone','p1'); insert into users values('usertwo','p2'); insert into users values('userthree','p3'); insert into groups values('userone','admin'); insert into groups values('usertwo','admin'); insert into groups values('userthree','user');

Create connection pool

- In the **Console Navigation** menu on the left click on **Database Pools**.
- Click on **Using the Geronimo database pool wizard**.
- Enter **SecurityDatabasePool** as the database pool name.
- Select **Derby embedded XA** from the database pool type pull-down menu and click **Next**.
- From the Driver JAR scroll box select **org.apache.geronimo.configs/system-database/2.1.1-SNAPSHOT/car**.
- Leave **blank** the DB user name and password.
- Enter **SecurityDatabase** as the database name.
- Click **Deploy**.

Add a new security realm

To create a new security realm click on **Add new security realm** from the **Security Realms** portlet.



Enter **derby_security_realm** in the **Name of Security Realm:** field and select **Database (SQL) Realm** from the **Realm type:** pull-down menu and click **Next**.

The following screen configures the login module. The first two field you need to fill may vary from one database type to another. In this case we are using the embedded Derby database so the User and Group select SQL should read as follows:

User SELECT SQL: `select username, password from users where username=?`
Group SELECT SQL: `select username, groupname from groups where username=?`

Once you entered the SQL statements for retrieving users and groups you need to select from the **Database Pool** pull-down menu the database connection pool you created in the previous step. Add the required values as shown below and click **Next**.

Database Pool: `SecurityDatabasePool`

Security Realms

Create Security Realm -- Step 2: Configure Login Module

User SELECT SQL: select username, password from users where username=?

A SQL statement to load user/password information. It should return 2 columns, the first holding a username and the second holding a password. The statement may use the PreparedStatement syntax of ? for a parameter, in which case the username will be set for every parameter. A typical setting would be `SELECT username, password FROM app_users WHERE username=?`

Group SELECT SQL: select username, groupname from groups where username=?

A SQL statement to load group information for a user. It should return 2 columns, the first holding a username and the second holding a group name. The statement may use the PreparedStatement syntax of ? for a parameter, in which case the username will be set for every parameter. A typical setting would be `SELECT username, group_name FROM user_groups WHERE username=?` or for a more normalized schema, `SELECT u.username, g.name FROM app_users u, groups g, user_groups ug WHERE ug.user_id=users.id AND ug.group_id=g.id AND u.username=?`

Digest Algorithm:

Message Digest algorithm (e.g. MD5, SHA1, etc.) used on the passwords. Leave this field empty if no digest algorithm is used.

Digest Encoding:

Encoding to use for digests (e.g. hex, base64). This is used only if a Message Digest algorithm is specified.If no encoding is specified, hex will be used.

A SQL security realm must either have a database pool or JDBC connectivity settings to connect to the database. Please select EITHER the database pool, OR the rest of the JDBC settings.

Database Pool SecurityDatabasePool

A database pool that the login module will use to connect to the database. If this is specified, none of the rest of the settings after this are necessary.

JDBC Driver Class

The fully-qualified JDBC driver class name. This driver must be located in the JAR specified in the next field.

Driver JAR:

The JAR holding the selected JDBC driver. Should be installed under GERONIMO/repository/ to appear in this list.

JDBC URL

The JDBC URL that specifies the details of the database to connect to. This has a different form for each JDBC driver.

JDBC Username

The username used to connect to the database

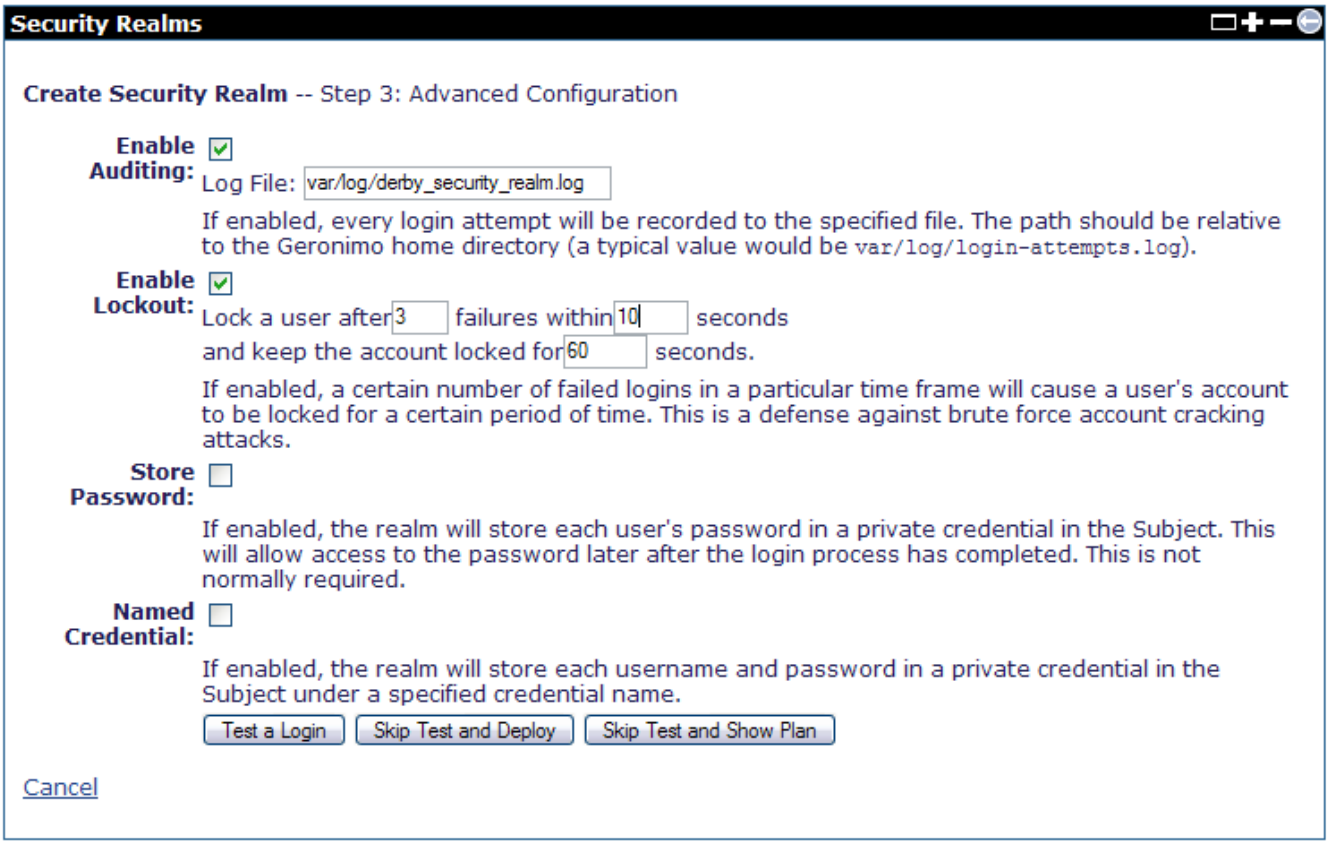
JDBC Password

Confirm Password

The password used to connect to the database

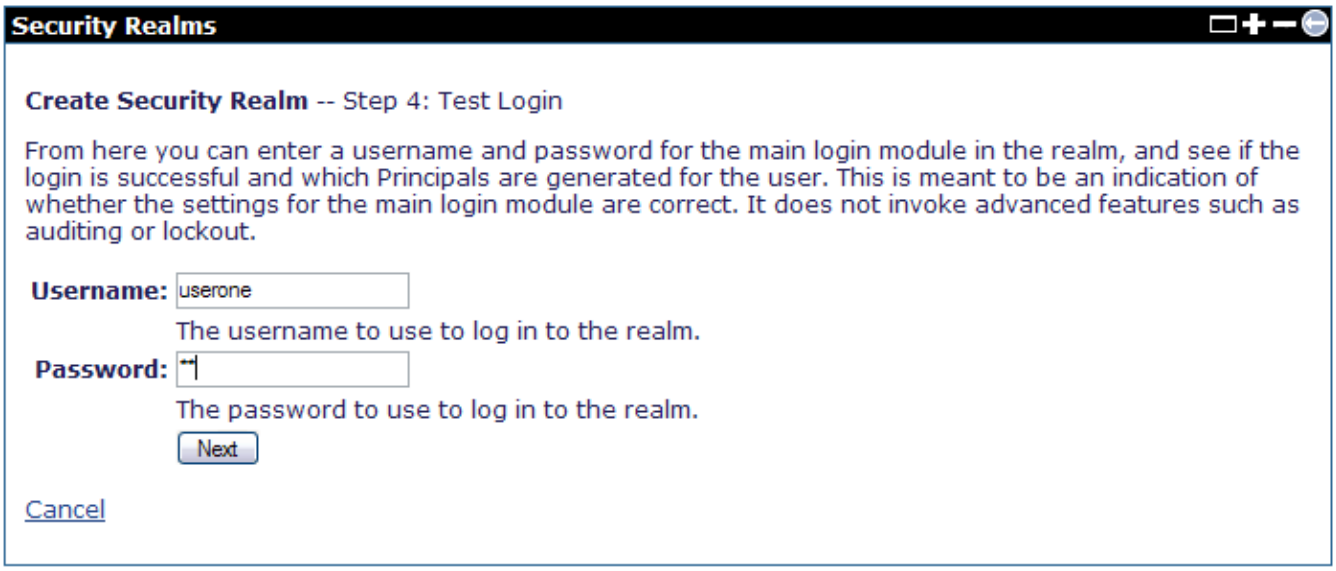
Next

The following step will allow you to enable auditing for monitoring the login attempts via this realm. In this step you can also configure the account lockout based on the number of failed loging attempts withing a specified timeframe. If you enable **Store Password**, then it will allow the realm to store the user's password in a private credential in the "Subject". If you enable **Naming Credential**, in addition to the user's password, this option will use private credentials to store user names too.

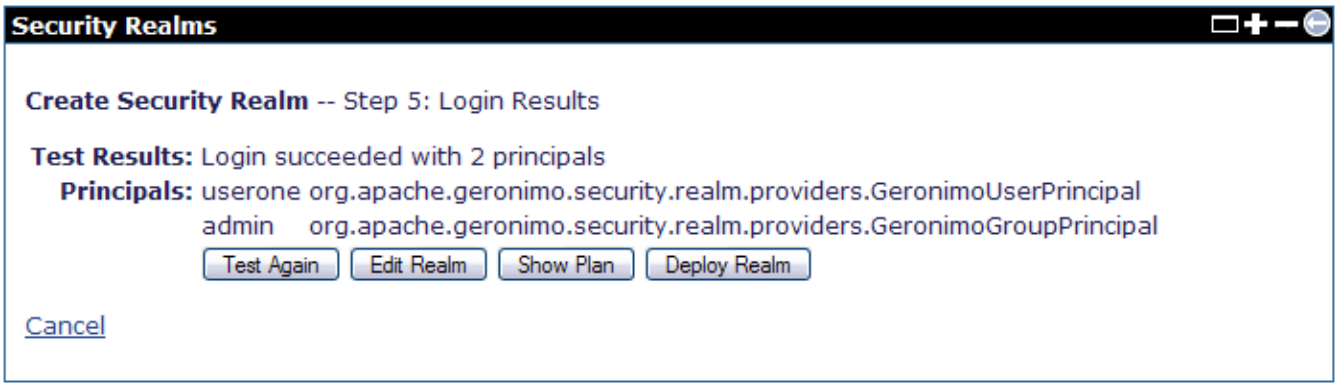


At this point you have configured this new security realm, the next step i to test it and then deploy it. Click on **Test a Login**.

Enter a valid user name and password to be retrieved from the database and click **Next**.



You should receive a confirmation message that the login succeeded, click on **Deploy Realm** to load this configuration to the server.



Now you have a new, fully configured, security realm that retrieves user names and passwords from the build in Derby database.

If you get an error the first time you try to validate this realm, you will very likely see the **`SQL Exception: Failed to start database ...`** error in the terminal and logs. This is a know issue with Derby, you will need to restart Geronimio so the new database can communicate properly.

The following example shows the deployment plan for this security realm. As an alternative to the Geronimo Administration Console, you can save this example to a file (i.e. derby_security_realm.xml) and deploy it with the Deployer tool by running the following command:

solid <geronimo_home>\bin\deploy --user system --password manager deploy derby_security_realm.xml xmlsolidderby_security_realm <module xmlns="http://geronimo.apache.org/xml/ns/deployment-1.2"> <environment> <moduleId> <groupId>console.realm</groupId> <artifactId>derby_security_realm</artifactId> <version>1.0</version> <type>car</type> </moduleId> <dependencies> <dependency> <groupId>org.apache.geronimo.framework</groupId> <artifactId>j2ee-security</artifactId> <type>car</type> </dependency> <dependency> <groupId>console.dbpool</groupId> <artifactId>SecurityDatabasePool</artifactId> <version>1.0</version> <type>car</type> </dependency> </dependencies> </environment> <gbean name="derby_security_realm" class="org.apache.geronimo.security.realm.GenericSecurityRealm" xsi:type="dep:gbeanType" xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <attribute name="realmName">derby_security_realm</attribute> <attribute name="global">true</attribute> <reference name="ServerInfo"> <name>ServerInfo</name> </reference> <xml-reference name="LoginModuleConfiguration"> <log:login-config xmlns:log="http://geronimo.apache.org/xml/ns/loginconfig-2.0"> <log:login-module control-flag="REQUIRED" wrap-principals="false"> <log:login-domain-name>derby_security_realm</log:login-domain-name> <log:login-module-class>org.apache.geronimo.security.realm.providers.SQLLoginModule</log:login-module-class> <log:option name="dataSourceName">SecurityDatabasePool</log:option> <log:option name="dataSourceApplication">null</log:option> <log:option name="groupSelect">select username, groupname from groups where username=?</log:option> <log:option name="userSelect">select username, password from users where username=?</log:option> </log:login-module> <log:login-module control-flag="OPTIONAL" wrap-principals="false"> <log:login-domain-name>derby_security_realm-Audit</log:login-domain-name> <log:login-module-class>org.apache.geronimo.security.realm.providers.FileAuditLoginModule</log:login-module-class> <log:option name="file">var/log/derby_security_realm.log</log:option> </log:login-module> <log:login-module control-flag="REQUISITE" wrap-principals="false"> <log:login-domain-name>derby_security_realm-Lockout</log:login-domain-name> <log:login-module-class>org.apache.geronimo.security.realm.providers.RepeatedFailureLockoutLoginModule</log:login-module-class> <log:option name="failureCount">3</log:option> <log:option name="failurePeriodSecs">10</log:option> <log:option name="lockoutDurationSecs">60</log:option> </log:login-module> </log:login-config> </xml-reference> </gbean> </module>

Once the security realm has been created, you can use the **usage** link to view samples of how to use the new realm in your applications.