HITACHI
Inspire the Next

Migrate Users/Roles from Pentaho Security to JDBC Security

HITACHI
Inspire the Next

Change log (if you want to use it):

Date	Version	Author	Changes

Contents

Overview

Many customers start out using Pentaho Security to secure their Pentaho server repositories. As the manually created number of users and roles grow over time, customers eventually reach a point where their implementations are no longer viable or supported (approx. 80-100 users/roles).

After you finish migrating your users, roles, and associations from Pentaho Security, you will no longer be using the Pentaho User Console (PUC) to manage security. This document demonstrates how to extract existing users, roles, and role-association data from Pentaho Security using Pentaho Data Integration (PDI) and loading it into Java Database Connectivity (JDBC) security tables. The process can be adapted to other advanced security options.

The intention of this document is to speak about topics generally; however, these are the specific versions covered here:

Software	Version(s)
Pentaho	6.x, 7.x, 8.0

The Components Reference in Pentaho Documentation has a complete list of supported software and hardware.

Migrating to JDBC Security

To migrate to JDBC security, follow these steps. You can find details on these topics in the following sections:

- Build JDBC Security Tables
- Migrate Data from Pentaho Security
- Configure the BA Server for JDBC Security
- Continue to Manage Security Data

Build JDBC Security Tables

The first step to migrating users, roles, and user data is to build the database tables to maintain the data. Three tables are required: `users`, `authorities`, and `granted_authorities`. The Oracle Data Definition Language (DDL) required to build these tables is listed below. If Oracle is not your database, then you will need to make minor modifications.

```
CREATE TABLE USERS(
USERNAME VARCHAR2(50) NOT NULL PRIMARY KEY,
PASSWORD VARCHAR2(50) NOT NULL,
ENABLED char(1) DEFAULT '1' NOT NULL,
DESCRIPTION VARCHAR2(100),
LOAD_DATE DATE DEFAULT SYSDATE
);

CREATE TABLE AUTHORITIES
(AUTHORITY VARCHAR2(50) NOT NULL PRIMARY KEY,
DESCRIPTION VARCHAR2(100),
LOAD_DATE DATE DEFAULT SYSDATE
);

CREATE TABLE GRANTED_AUTHORITIES
(USERNAME VARCHAR2(50) NOT NULL,
AUTHORITY VARCHAR2(50) NOT NULL,
LOAD_DATE DATE DEFAULT SYSDATE,
CONSTRAINT PK_GRANTED_AUTH primary key (USERNAME,AUTHORITY),
CONSTRAINT FK_GRANTED_AUTH_USERS FOREIGN KEY(USERNAME) REFERENCES
USERS(USERNAME) DISABLE NOVALIDATE,
CONSTRAINT FK_GRANTED_AUTH_AUTH FOREIGN KEY(AUTHORITY) REFERENCES
AUTHORITIES(AUTHORITY) DISABLE NOVALIDATE
);
```

Migrate Data from Pentaho Security

PDI can be used to extract security data from Pentaho Security and load the new tables, once the database tables are built.

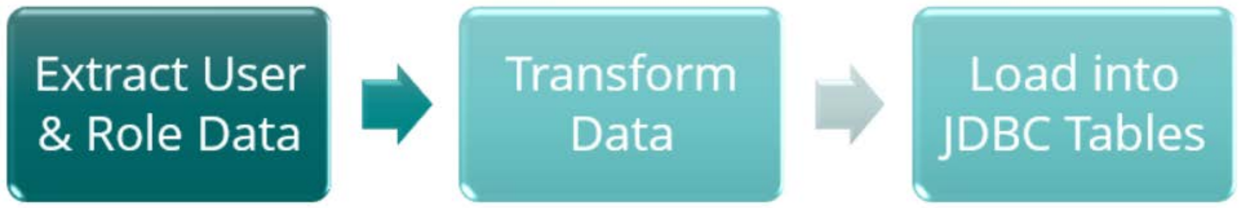

Figure 1: Data Migration

If you intend to use another supported security option instead of JDBC tables, you can modify the sample PDI transformation to write to a different format.

This document uses examples based on a sample file to help illustrate the processes. Make sure you assign default passwords to your users. This sample, while not supported by Pentaho, can serve as a template for you to use in your environment. Such efforts should be validated in a test environment prior to advancing to a production environment.

The sample PDI transformation extracts the data from Pentaho Security by using the `UserRoleListResource` service. This Pentaho service lists roles, permissions, and users. The PDI **Rest Client** step is used to retrieve the list of roles, permissions, and users. Other PDI steps are used to transform and load this data into the JDBC tables.

After you finish migrating your users, roles, and associations from Pentaho Security, you will no longer be using the Pentaho User Console to manage security.

In order to make the data migration process flexible, the following parameters must be provided:

- **db_name** – The database that maintains the new security tables.
- **db_host_name** – The database host name.
- **db_port_number** – The database port number.
- **db_user_name** – The database user. This user must have the appropriate privileges to insert data into the new tables. You must add schema values to the table output steps or synonyms in the database if this user is not the table owner.
- **db_password** – The database user password.
- **pentaho_server_url** – The server url + api path. For example, `http://localhost:8080/pentaho/api/userroledao`.

Configure the Server for JDBC Security

After you have validated that your JDBC tables have data, you can make changes to your server configuration files to switch from using Pentaho Security to JDBC security. Manual JDBC Connection Configuration steps are available in the Pentaho documentation.

Continue to Manage Security Data

After moving away from Pentaho Security, users, roles, and user role data are no longer managed by the Pentaho User Console (PUC). Therefore, you will need to develop a method to manage the data in these tables. A simple web application, a transformation that loads data from a file (such as a .CSV or spreadsheet), or an open-source SQL editor, are all viable options.

Related Information

Here are some links to information that you may find helpful while using this best practices document:

- Pentaho Developers Center - User Role List Resource
- Manual JDBC Connection Configuration
- Pentaho Components Reference