*HPC Data MANAGEMENT*

DEVELOPMENT ENVIRONMENT SETUP

Version *1.5*

*12/03/2017*

**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version Number** | **Implemented**  **By** | **Revision**  **Date** | **Description of Change** |
| 1.0 | Eran Rosenberg | *5/17/2015* | Initial Draft |
| 1.1 | Eran Rosenberg | 8/17/2015 | Create local Keystore |
| 1.2 | Eran Rosenberg | 1/1/2017 | Updates after lib/tools upgrades |
| 1.3 | Eran Rosenberg | 6/12/2017 | Configure servicemix w/ Keystore + Git repo |
| 1.4 | William Liu | 11/16/2017 | Added material in Build section about saving time on Maven builds |
| 1.5 | Eran Rosenberg | 12/17/2017 | Added section to configure Eclipse and Code Style formatter |

TABLE OF CONTENTS

[Git Code Repository 4](#_Toc500057067)

[Java Development Kit (JDK) 4](#_Toc500057068)

[MAVEN 4](#_Toc500057069)

[SERVICEMIX 4](#_Toc500057070)

[PostgreSQL DB 5](#_Toc500057071)

[IRODS 5](#_Toc500057072)

[PATH Environment Variable 5](#_Toc500057073)

[BUILD 5](#_Toc500057074)

[DEPLOY API to SERVICEMIX 6](#_Toc500057075)

[ENDPOINTS 6](#_Toc500057076)

[SOAP-UI 6](#_Toc500057077)

[Eclipse 7](#_Toc500057078)

# Git Code Repository

Check out the source code from Git:

Repository URL: <https://github.com/CBIIT/HPC_DME_APIs>

Branch name: master

Set HPC\_HOME environment variable to the ‘hpc/src’ directory path in the source tree.

e.g /Development/HPC-DM/src/hpc

# Java Development Kit (JDK)

Install JDK 1.8.x: <http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html>

Set JAVA\_HOME environment variable accordingly.

# MAVEN

Install Maven 3.3.9: <http://maven.apache.org/download.cgi>

Set MAVEN\_HOME environment variable accordingly.

# SERVICEMIX

Install Servicemix 7.0.0.: <http://servicemix.apache.org/downloads.html>

Set SERVICE\_MIX\_HOME environment variable accordingly.

To allow the server to communicate with iRODS and LDAP, we need to deploy/configure a keystore to Servicemix:

1. Download the keystore file named hpc-keystore.jks from the GitHub repository at <https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/development/hpc-keystore.jks>. Alternatively, find this keystore file in your local clone of the GitHub repository.
2. Copy hpc-keystore.jks to $SERVICE\_MIX\_HOME/etc/ directory.
3. Add the following to the end of $SERVICE\_MIX\_HOME/etc/system.properties

# HPC-DM keystore

javax.net.ssl.keyStore=${karaf.home}/etc/hpc-keystore.jks

javax.net.ssl.keyStorePassword=hpc-server-store-pwd

javax.net.ssl.trustStore=${karaf.home}/etc/hpc-keystore.jks

javax.net.ssl.trustStorePassword=hpc-server-store-pwd

# PostgreSQL DB

Install PostgreSQL DB 9.6.3 <https://www.postgresql.org/download/>

1. Execute all SQL scripts in $HPC\_HOME/hpc-server/hpc-dao-impl/src/main/scripts/schema **except** 
   1. hpc\_hierarchical\_metadata.sql (this script is depending on having iRODS installed which we don’t have on local development environment).
   2. hpc\_system\_account.sql (The local-dev version will create foreign tables, so you don’t need to maintain system account credentials locally)
2. Execute SQL scripts intended for DEV environment (they end with ‘dev’ in the script name’) in $HPC\_HOME/hpc-server/hpc-dao-impl/src/main/scripts/restore.
3. Manually insert a row into HPC\_USER table using your info (NIH user-id, first-name, last-name, etc)

# IRODS

You will need an iRODS account in the DEV environment. Ask a team member to register you as an HPC-DM user in the DEV environment. Use your NIH user-id and first/last name. The registration in DEV will create your iRODS user.

# PATH Environment Variable

Set your $PATH environment variable

PATH=$MAVEN\_HOME/bin:$SERVICE\_MIX\_HOME/bin:$PATH

# BUILD

cd $HPC\_HOME

mvn clean install

Saving time on Maven builds

The above instruction runs Maven build of **all** HPC DME software products. This could be overkill, and consequently you spend more time executing a build than necessary due to building products that you do not need.

To save time on Maven builds, invoke Maven using the -pl option to exclude modules from the build process. Two Maven command line invocations applying the -pl option are shown next as examples.

Windows command line invocation, notice double quotes for -pl switch argument

>mvn **-pl “!hpc-server/hpc-ws-rs-test,!hpc-cli”** clean install

Unix/Linux/Mac command line invocation, notice single quotes for -pl switch argument

>mvn **-pl** **‘!hpc-server/hpc-ws-rs-test,!hpc-cli’** clean install

With the -pl option, you can indicate which modules to skip by specifying a comma-separated list of module specifications each prefixed with !.  In the above examples, the **hpc-ws-rs-test** module of the **hpc-server** module and the **hpc-cli** module are declared to be skipped.

The preceding examples assume you want to build all modules except for specific exclusion(s). If instead you want to build specific module(s) and exclude the rest, you can use the -pl option to specify comma-separated list of modules you wish to build. In that case, each module is declared without the ! prefix.

# DEPLOY API to SERVICEMIX

Find the server version (from the top level pom file). You will need to use it in the command to start the server.

Run Servicemix console: servicemix

Install hpc-server (in Servicemix console):

features:repo-add mvn:gov.nih.nci.hpc/hpc-features*/<server-version>/*xml/features

feature:install hpc-server-rest-services

feature:install hpc-server-scheduler

Note: in a local development environment, we typically don’t run the scheduler since we share the same iRODS server with DEV and the scheduler is running there. Having 2 scheduler running (pointing to the same iRODS but having different HPC DB) will cause issues in the async upload of data objects.

Also – you can start servicemix in a debug mode by ‘servicemix debug’, and then attach a debugger to a remote Java application on port 5005.

# ENDPOINTS

HPC DM Services are deployed to [https://localhost:7338/hpc-server/<hpc-service](https://localhost:7338/hpc-server/%3chpc-service)>

# SOAP-UI

Soap UI workspace and project can be found under $HPC\_HOME/hpc-soap-ui folder.

Note – after pointing soap-ui to the workspace, you will need to correct the path to the project based on the directory you pulled the code into.

# Eclipse

To create Eclipse projects:

cd $HPC\_HOME/hpc-server

mvn eclipse:eclipse

cd $HPC\_HOME/hpc-web

mvn eclipse:eclipse

You can now import the HPC-DME projects into Eclipse. Make sure it builds clean within Eclipse.

We use the Google Code Style for Java. To set up eclipse, we will use 2 files found under:

<https://github.com/CBIIT/HPC_DME_APIs/blob/master/doc/development/hpc-code-style>

First copy the jar file to your Eclipse ‘dropins’ folder. Then restart Eclipse.

Go to Preferences -> Java -> Code Style -> Formatter

Import the XML file as your profile, and select the google formatter.

