

# LAB- Securing properties

In this Lab, you will understand how to encrypt your externalized properties

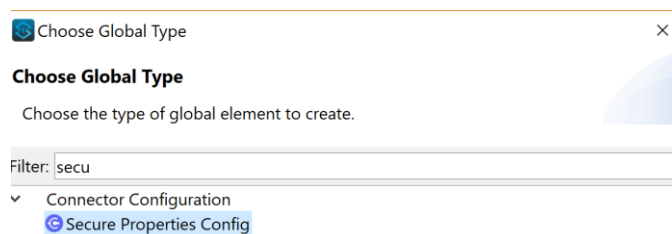
In this lab, you will be working on **Same Project** which u worked on for externalization exercise

1) We need to add “Mule Secure Configuration” module now. It is not available in Anypoint studio by default. We need to add it from anypoint exchange.

Click on “Search in exchange” button in the Mule palette on right side.

Then search for “Secure” and Select “Mule Secure Configuration” module.

Now, Select Global elements tab in your XML and click on “Create”. Then select “Secure properties Config” as shown below :



Now Configure Secure Properties Config as shown below :


**Secure Properties Config**

General Advanced Notes Help


Name: Secure\_Properties\_Config

General


File: db-secure-dev.yaml


Key:  abcdefghijklmnop

File level encryption: False (Default)

Encoding: 

encrypt

Algorithm:  Blowfish

Mode:  CBC (Default)

Now we have to generate db-secure-dev.yaml file with encrypted property values

I have given a jar file with name secure-properties-tool.jar. This jar can also be downloaded from following documentation page of mulesoft

**<https://docs.mulesoft.com/mule-runtime/4.5/secure-configuration-properties>**

We can use command like below :

```
java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool  
<method> <operation> <algorithm> <mode> <key> <input file> <output file>
```

create a folder with name “secure-properties” anywhere on your machine.

Under that folder copy the secure-properties-tool.jar and db-dev.yaml file

Open cmd in this secure-properties folder and execute the below command (Type this command. If u copy paste, there might be a problem.)

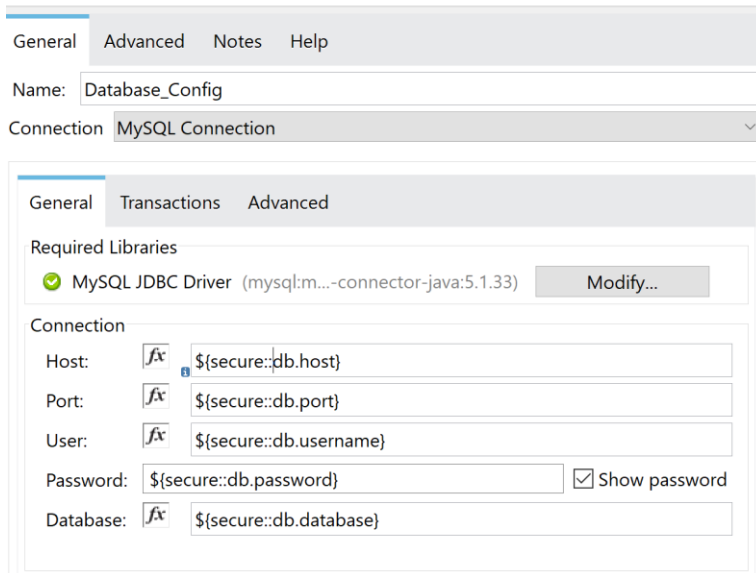
```
java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool  
file encrypt Blowfish CBC mulesoft db-dev.yaml db-secure-dev.yaml
```

Now, copy the generated db-secure-dev.yaml into src/main/resources.

Now, edit the Database Config as shown below :

## Database Config

Default configuration



General Advanced Notes Help

Name: Database\_Config

Connection: MySQL Connection

General Transactions Advanced

Required Libraries

MySQL JDBC Driver (mysql:mysql-connector-java:5.1.33) Modify...

Connection

Host: \${secure::db.host}

Port: \${secure::db.port}

User: \${secure::db.username}

Password: \${secure::db.password} ☒ Show password

Database: \${secure::db.database}

Now, go to Global elements tab and remove Configuration Properties element as it is no more required.

Run the application and give a request to <http://localhost:8081/db> and observe that there are no exceptions and the encrypted properties are picked correctly.

We can also have some properties in db-secure-dev.yaml which are not encrypted. For example, I don't want db.username to be encrypted. So, Change the value of db.username as "root" (in double quotes).

Run the application and observe that it still works same.

# This is the end of the Exercise