RECIPE 26

Mule Message Encryption and Decryption

The Encryption and Decryption of APIs allow to store information or to communicate with other parties while preventing uninvolved parties from understanding the stored information or understanding the communication.

The Encrypt Data API protects data privacy by scrambling clear data into an unintelligible form. To recover clear data from the encrypted data, use the Decrypt Data which will restore encrypted data to a clear (intelligible) form. Both processes involve a mathematical formula (algorithm) and secret data (key).

This recipe would boost your confidence level on MuleSoft Anypoint Platform for how easy and quick it is to encrypt and decrypt data. It can be achieved in many ways. This recipe uses Java Cryptology Extension (JCE), as part of the Java Cryptography Architecture (JCA) which encodes a message payload, or part of a payload.

GitHub location of this recipe:
https://github.com/WHISHWORKS/mule-api-recipes/tree/publish_v1.0/api-encryption

Pre-requisites

1. MuleSoft Anypoint Studio 6.1 or above (https://www.mulesoft.com/lp/dl/studio) installed
2. MuleSoft CloudHub Account (https://anypoint.mulesoft.com)
3. Application api-encryption is available on GitHub location of this recipe provided above

Process

High Level Steps:

A. Installing Anypoint Enterprise Security
B. Implementation of the API
C. Testing

A. **Installing Anypoint Enterprise Security**

1. Launch Anypoint Studio.

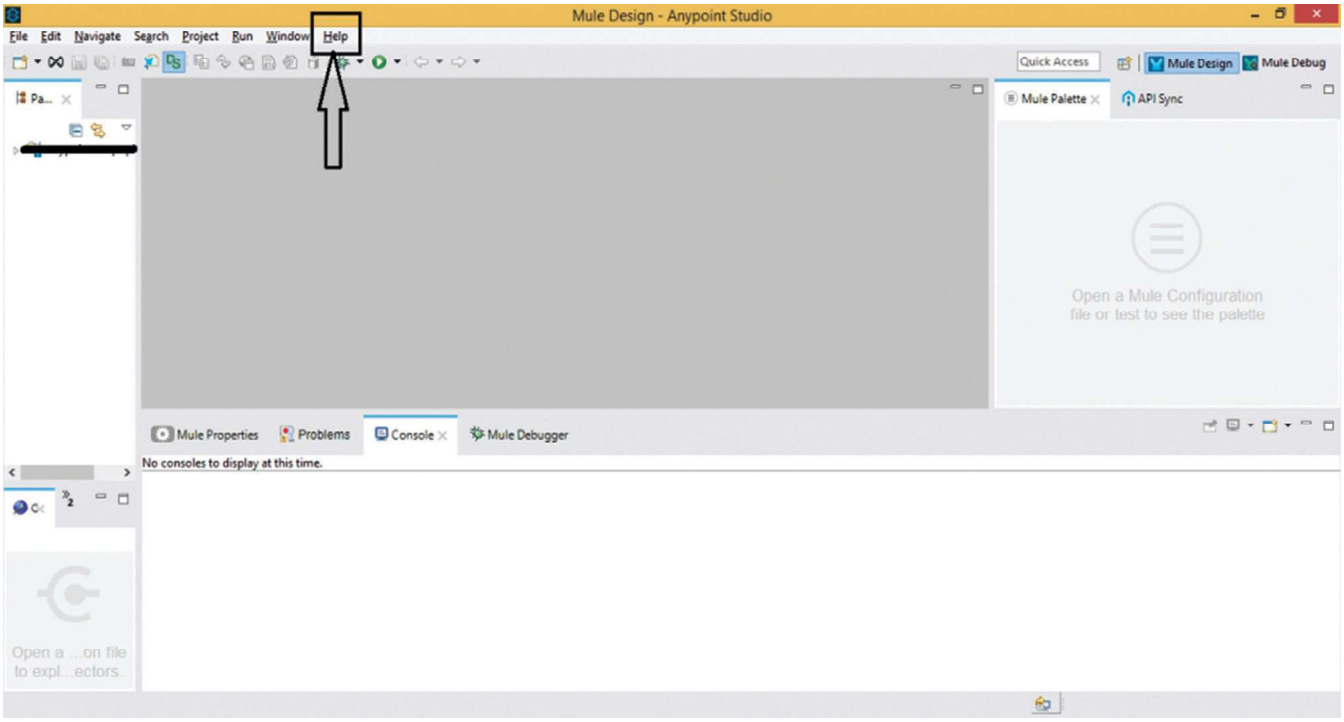

Figure 26.1: Anypoint Studio Home

2. Under the **Help** menu, select **Install New Software** as shown below:

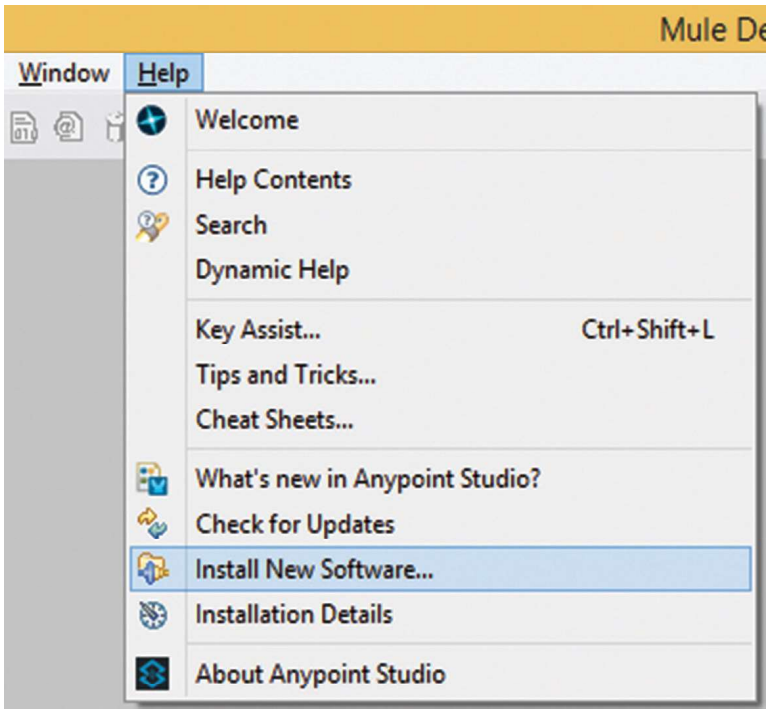

Figure 26.2: Anypoint Studio Help

3. Install wizard opens. Click the **Add** button next to the **Work with** field.

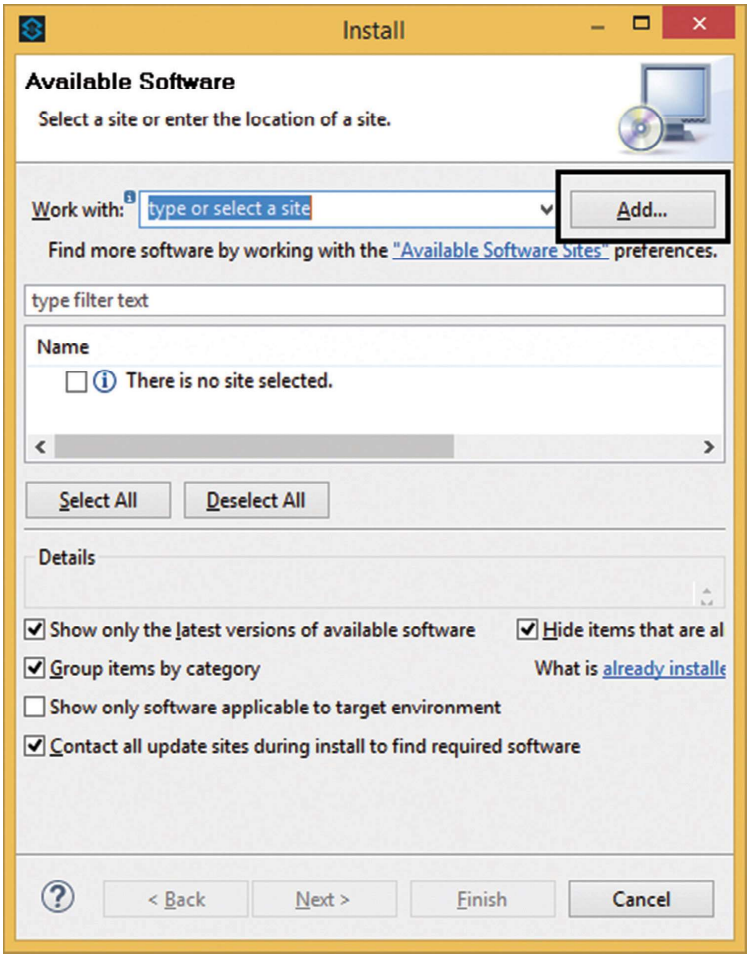

Figure 26.3: Install Software

4. In **Add Repository** panel, provide a **Name** for the repository as Anypoint Enterprise Security, and provide the link in the **Location** field as http://security-update-site-1.6.s3.amazonaws.com. Click **OK** as shown below:

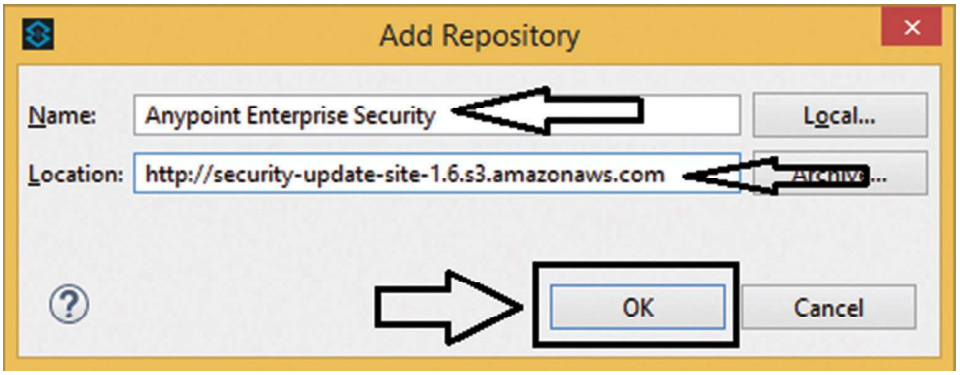

Figure 26.4: Add Repository

5. In the table, check the box to select **Premium**. Click **Next** as shown below:

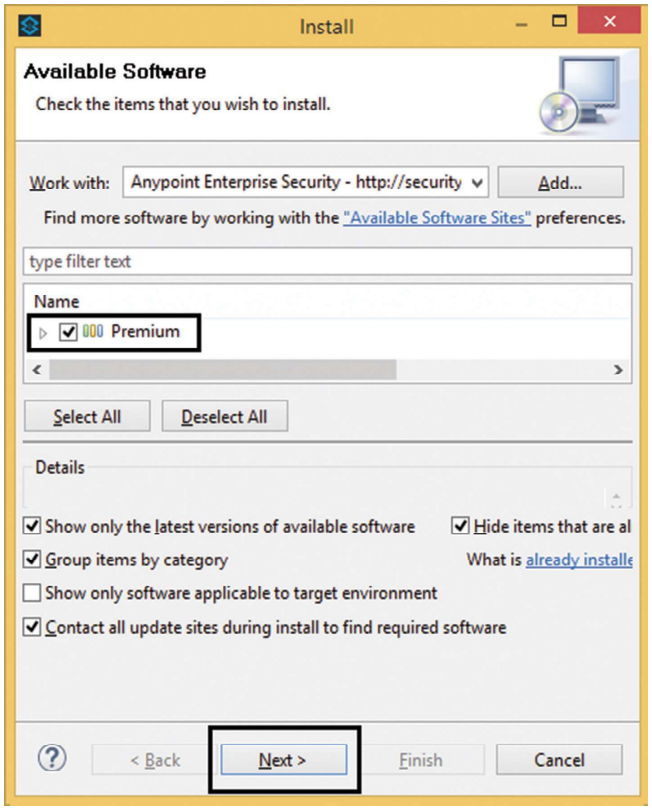

Figure 26.5: Install Wizard

6. Click **Next** in the next wizard pane.
7. Use the radio button to accept the terms of the license agreement, then click **Finish**.
8. Anypoint Studio installs **Anypoint Enterprise Security**, and asks to restart the application.
9. After Anypoint Studio Application's relaunch, Studio displays a new palette group called Security which contains six new message processors as shown below:

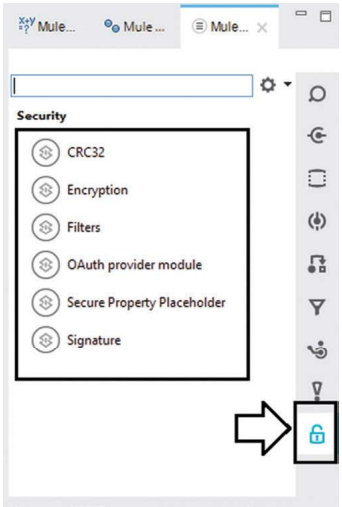

Figure 26.6: Palette - Security

B. **Implementation of the API**

1. Launch Mule Anypoint Studio and create a new **Mule Project** as shown below:
   File > New > Mule Project
2. In the **New Mule Project** dialogue that comes up, provide the **Project Name** as api-encryption. Click **Finish**.

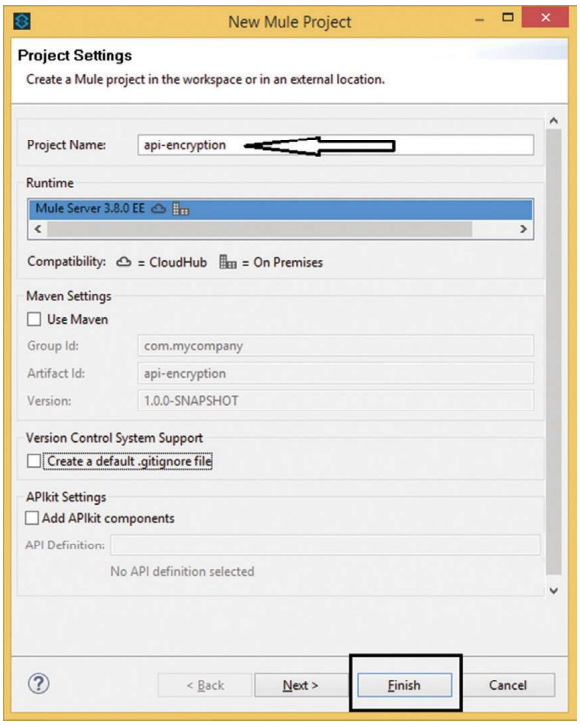

Figure 26.7: New Mule Project

3. This will create a blank project (as shown below) where Mule Flow will be implemented.

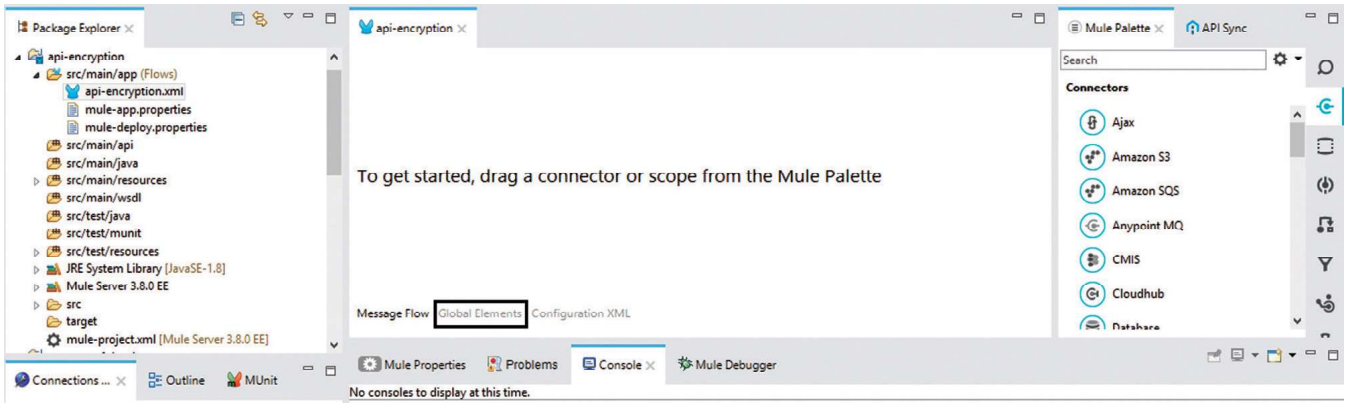

Figure 26.8: Project Structure

4. Click on **Global Elements** as shown above.
5. HTTP Connector should be added as a global element.
   a. Click on **Create** button.

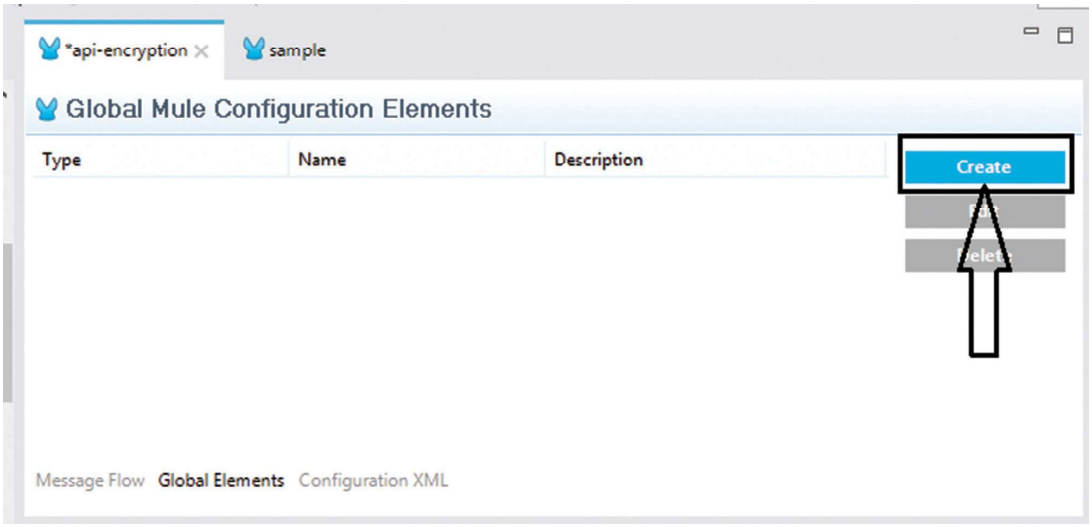

Figure 26.9: Global Elements Blank

b. **Choose Global Type** dialogue appears. Search for **http** in **Filter** field for HTTP Listener configuration connector.

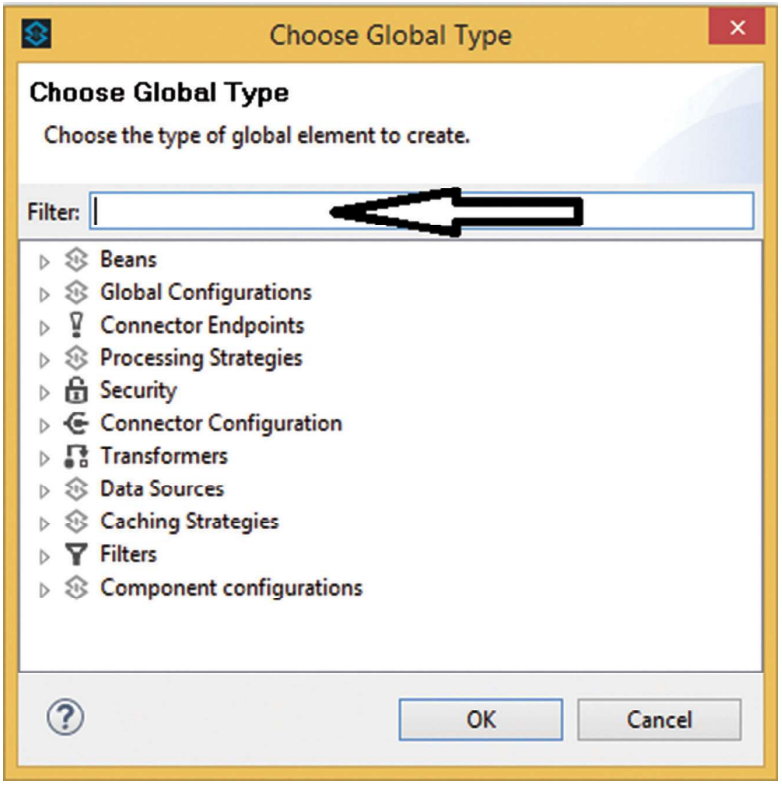

Figure 26.10: Choose Global Type

c. Select **HTTP Listener Configuration** that appears under Connector Configuration as shown below and click **OK.**

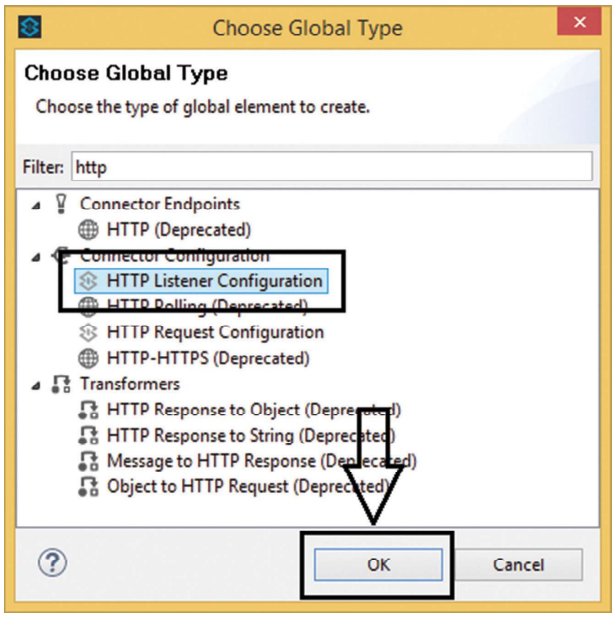

Figure 26.11: Global Element – HTTP Connector

d. **Global Element Properties** dialogue would be displayed. Click **OK.**

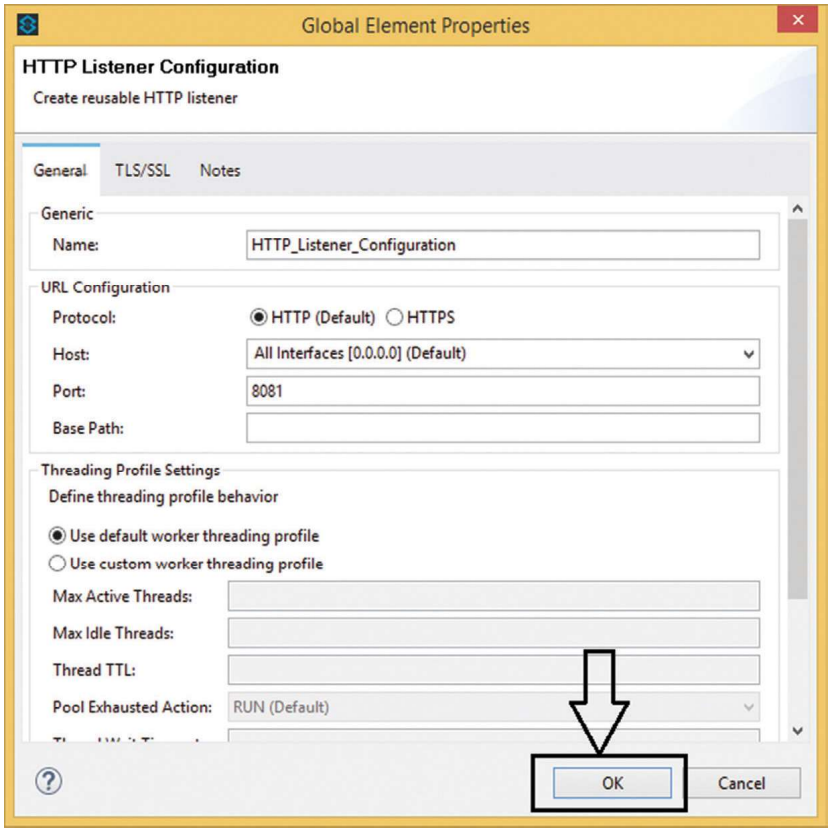

Figure 26.12: Global Element Properties - HTTP

e. The Global Element will be updated in **Global Element** tab as shown below:

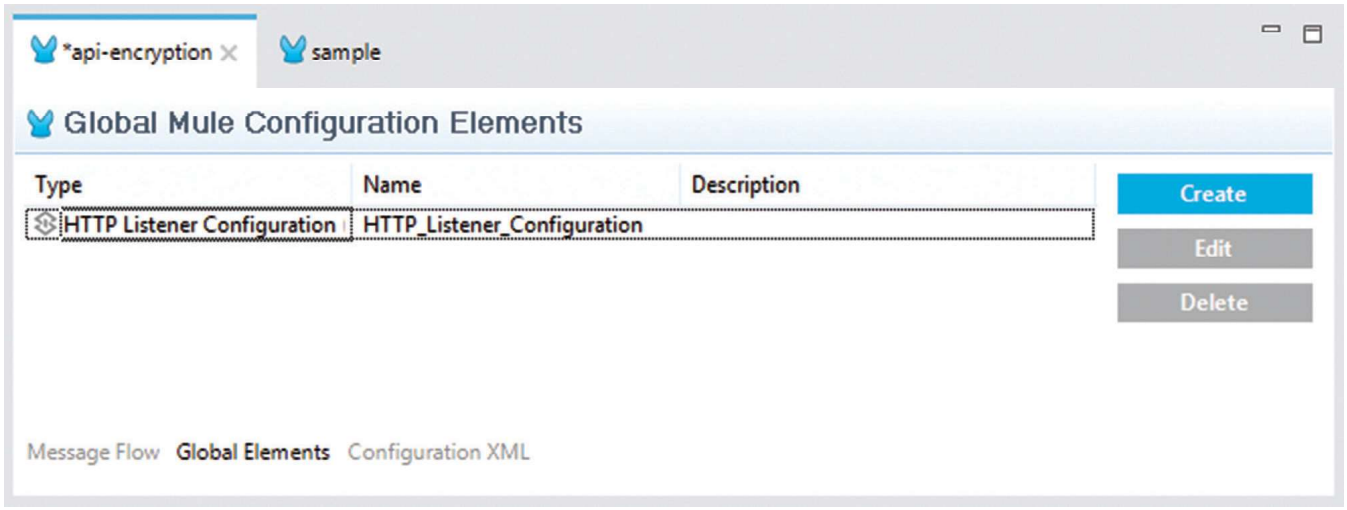

Figure 26.13: Global Elements - HTTP

6. Now, Encryption connector as a global element is also required for this project.
   a. Click on **Create** button
   b. Choose Global Type dialogue appears. Search for **encryption** in **Filter** field for Encryption connector.
   c. Select **Encryption** that appears under Connector Configuration as shown below and click **OK.**

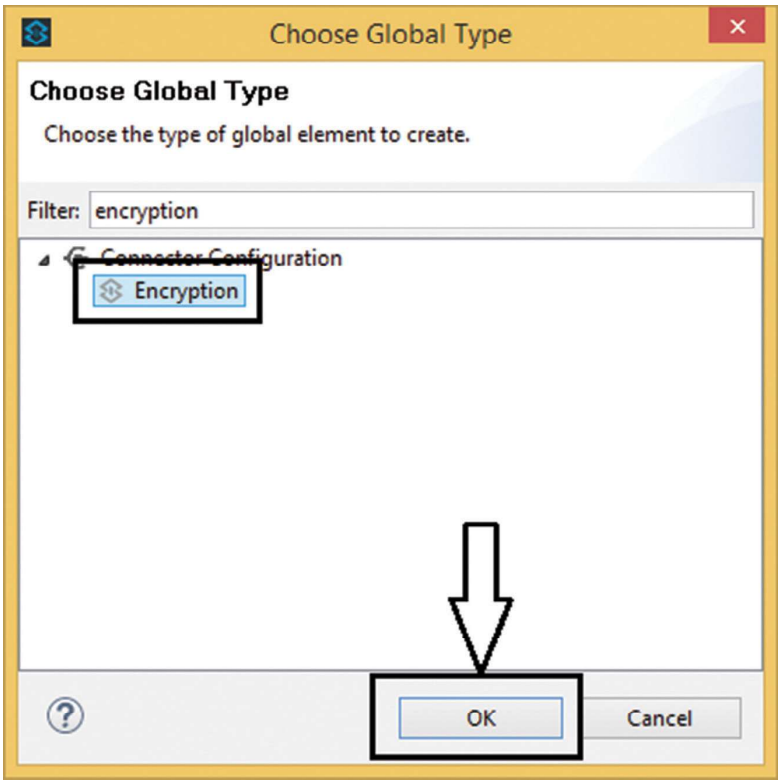

Figure 26.14: Global Element – Encryption Connector

d. **Global Element Properties** dialogue would be displayed.

Figure 26.15: Global Element Properties - Encrypter

e. Leave Default Encrypter as **JCE_ENCRYPTER(Default).** Click Jce Encrypter as shown above.
   i. Select **Define attributes** radio button:
   ii. Provide 16 bytes key i.e. kishankrsoni2512
   iii. Leave other fields as default
   iv. Click OK

Figure 26.16: Global Element Properties – JCE Encrypter

f. The Global element will be updated in **Global Element** tab as shown below:

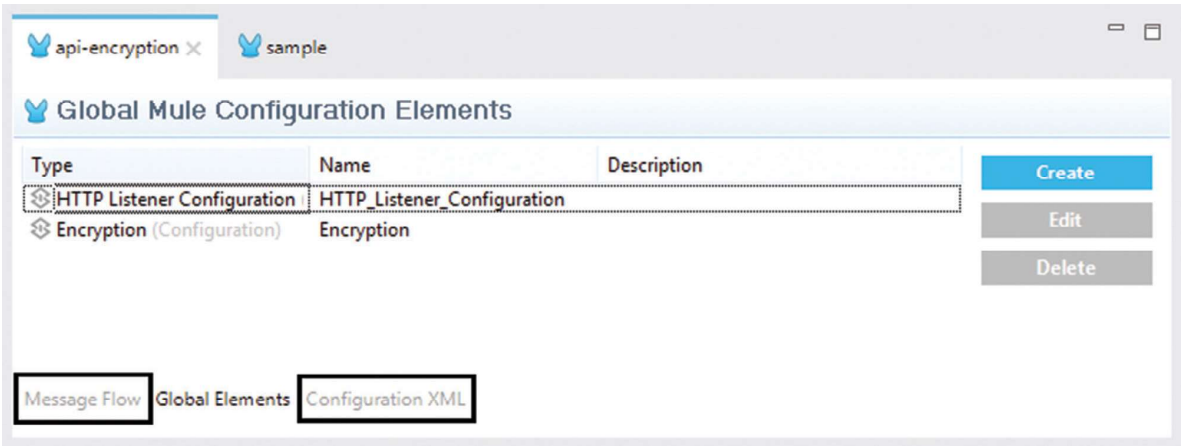

Figure 26.17: Global Elements

7. Go to **Package Explorer** and open **mule-app.properties** under this application **api-encryption** as shown below:

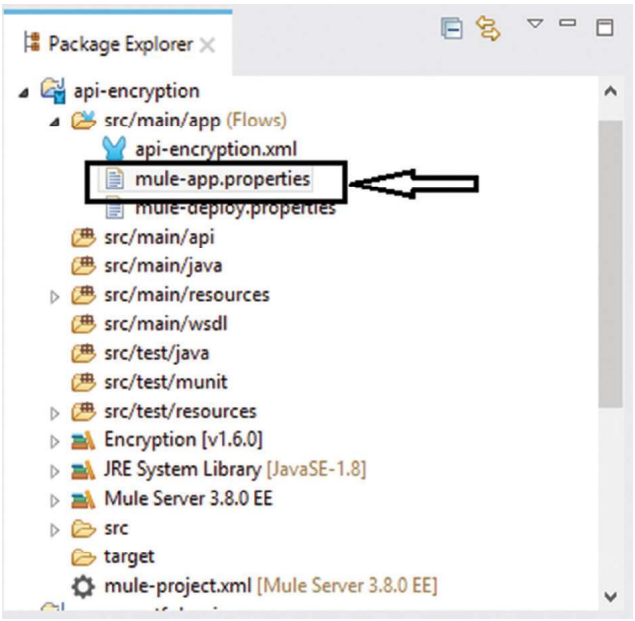

Figure 26.18: Application – Path Update

8. Provide the path for **path.encryptedFileLocation** and **path.output** as shown below:

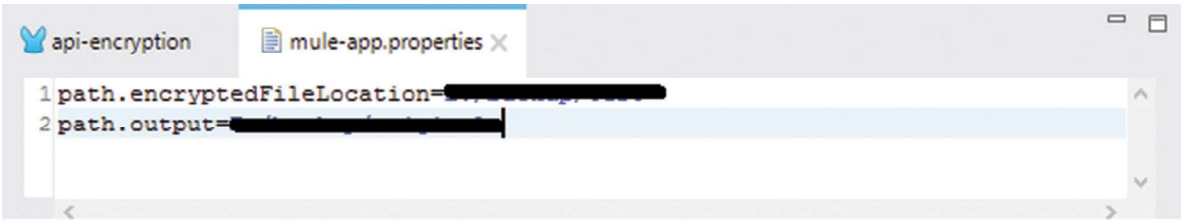

Figure 26.19: Mule-app.properties

9. Navigate to **Message Flow** tab.
   a. Drag **HTTP connector** from Mule Palette and drop it in Message Flow as shown below:

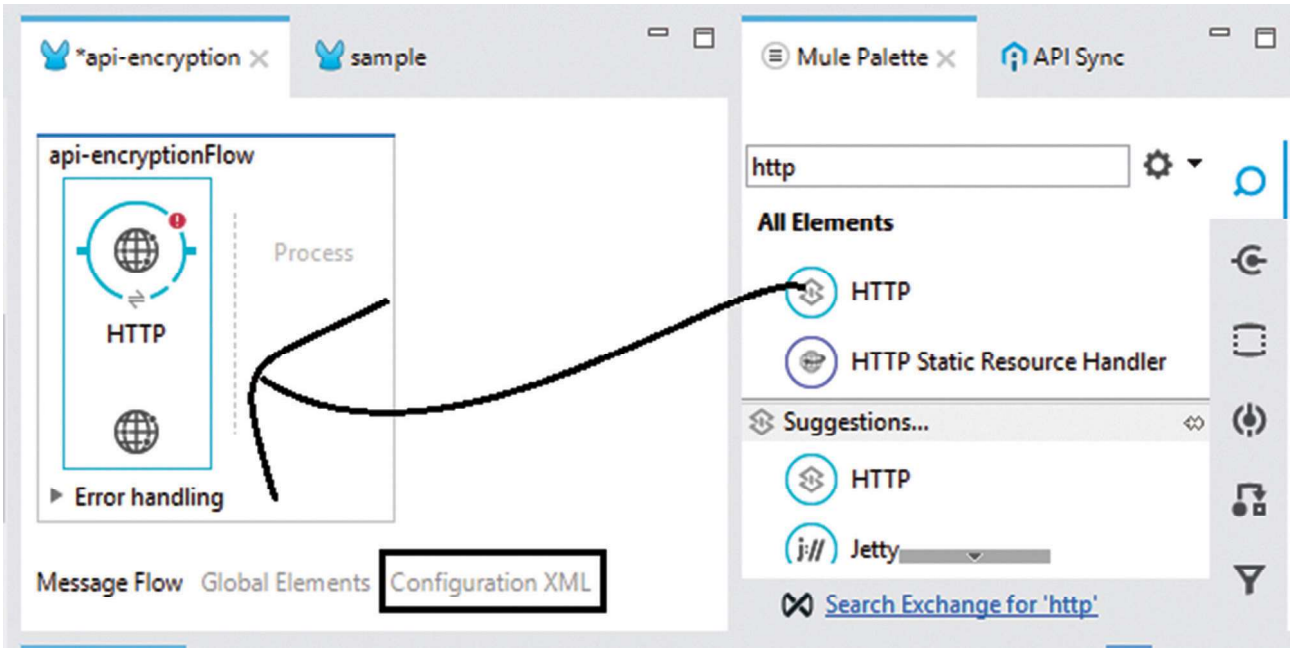

Figure 26.20: Message Flow - HTTP

   b. Click on the **Configuration XML** tab and search with the flow name i.e. *api-encryptionFlow flow*. The corresponding Mule Flow XML content should look like:

```
<flow name="api-encryptionFlow">
  <http:listener config-ref="HTTP_Listener_Configuration" path="/" doc:name="HTTP"/>
</flow>
```

10. Implement the HTTP method in *api-encryptionFlow* flow.
    a. In this Mule Flow replace the XML with the following XML content:

```
<flow name="api-encryptionFlow">
     <http:listener config-ref="HTTP_Listener_Configuration" path="/jceEncryptor"
doc:name="HTTP"/>
        <set-payload value="kishan" doc:name="Input"/>
            <encryption:encrypt  config-ref="Encryption"  using="JCE_ENCRYPTER"
doc:name="JCE Encrypter">
           <encryption:jce-encrypter key="kishankrsoni2512"/>
        </encryption:encrypt>
                 <file:outbound-endpoint  path="${path.encryptedFileLocation}"
outputPattern="sample.txt" responseTimeout="10000" doc:name="Writing into Text
File"/>
       <logger level="INFO" doc:name="Encrypted data Value"  message="#[payload]"/>
</flow>
```

In the code above, the request data is received and encrypted through JCE Encrypter and write into text file named as **sample.txt** at the given path provided in **mule-app.properties** as **path.encryptedFileLocation.**

b. **Save** the file and click on **Mule Flow** tab. The corresponding Mule Flow should now look like:

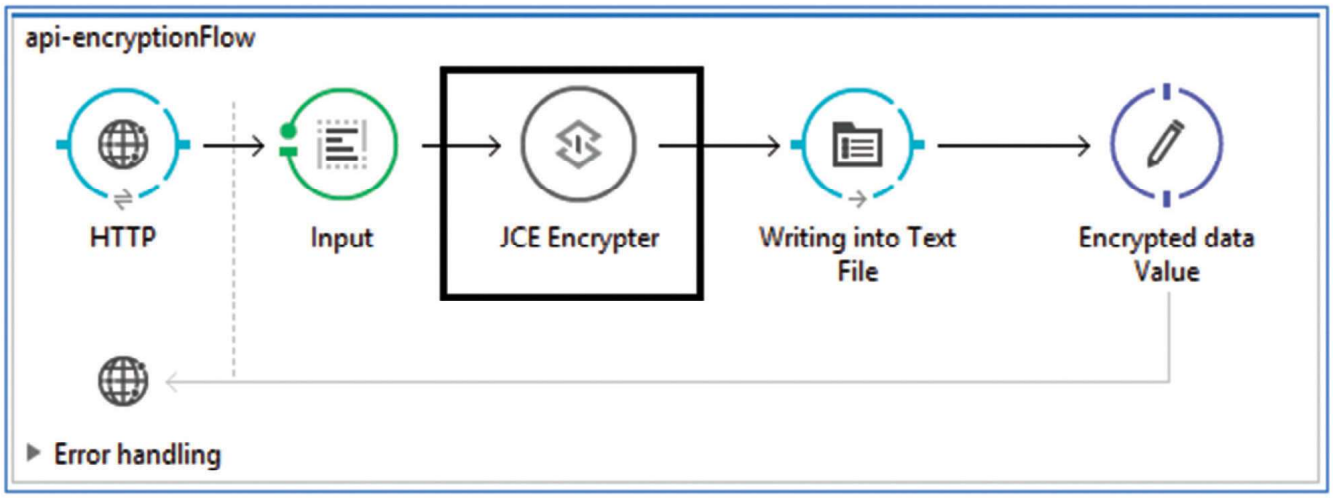

Figure 26.21: Encryption Flow

11. Now, it's time to develop a flow to decrypt encrypted text file and implement it.
12. Again, navigate to **Message Flow** tab.
    a. Drag **HTTP connector** from **Mule Palette** and drop it in the Message Flow as shown below:

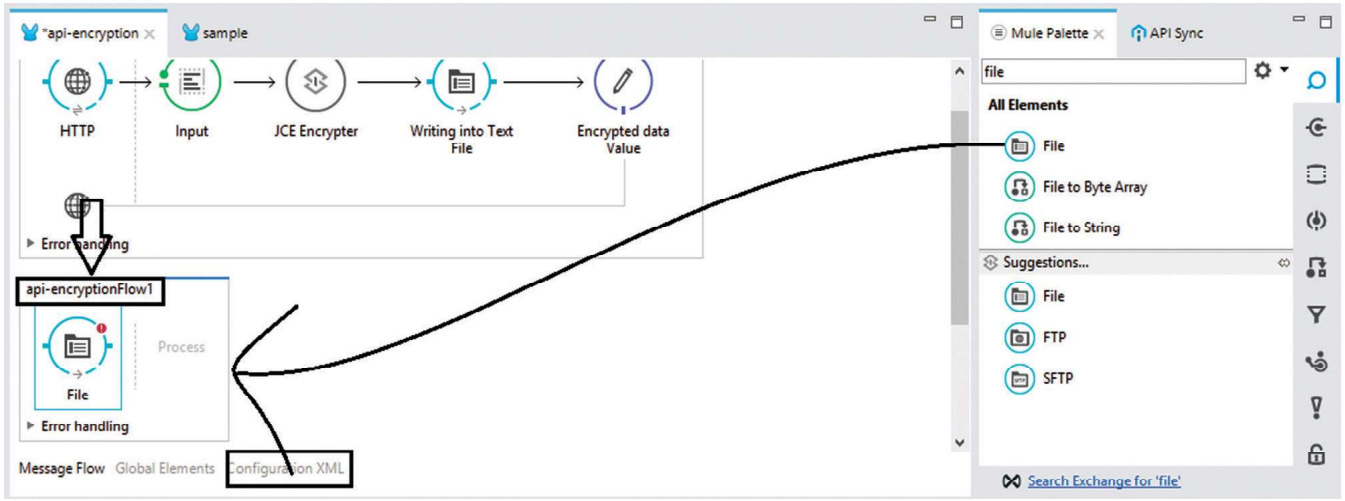

Figure 26.22: Message Flow - File

b. Change the flow name **api-encryptionFlow1** to **decryptionFlow** by clicking on **Flow name** as shown in above image.

c. Click on the **Configuration XML** tab and search with the flow name i.e. decryptionFlow flow. The corresponding Mule Flow XML content should look like:

```
<flow name="DecryptionFlow">
        <file:inbound-endpoint responseTimeout="10000" doc:name="File"/>
</flow>
```

13. Implement the HTTP method in *decryptionFlow* flow.
    a. In this Mule Flow, replace the XML with the following XML content:

```
<flow name="DecryptionFlow">
      <file:inbound-endpoint                path="${path.encryptedFileLocation}"
responseTimeout="10000" doc:name="Read data from text file">
            <file:filename-regex-filter pattern="sample.txt" caseSensitive="false"/>
      </file:inbound-endpoint>
      <file:file-to-string-transformer doc:name="File to String"/>
      <encryption:decrypt config-ref="Encryption" doc:name="Decryption">
            <encryption:jce-encrypter key="kishankrsoni2512"/>
      </encryption:decrypt>
      <set-payload value="#[payload] is a good boy" doc:name="Implementation of
decypted data"/>
      <file:outbound-endpoint  path="${path.output}"  outputPattern="final.txt"
responseTimeout="10000" doc:name="Write into file"/>
</flow>
```

    b. In the code above, the encrypted data will be read from the text file and decrypted through JCE decrypter and written into the text file named as **final.txt** at the given path provided in **mule-app.properties** as **path.output.**
    c. **Save** the file and click on **Mule Flow** tab. The corresponding Mule Flow should now look like:

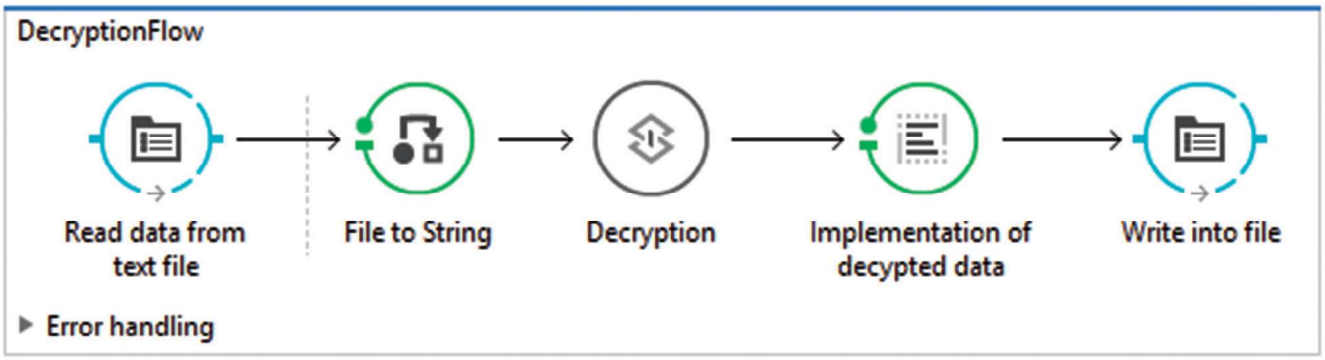

Figure 26.23: Decryption of encrypted data and its implementation

14. Right click on project name and select **Run As > Mule Application** to run this application.

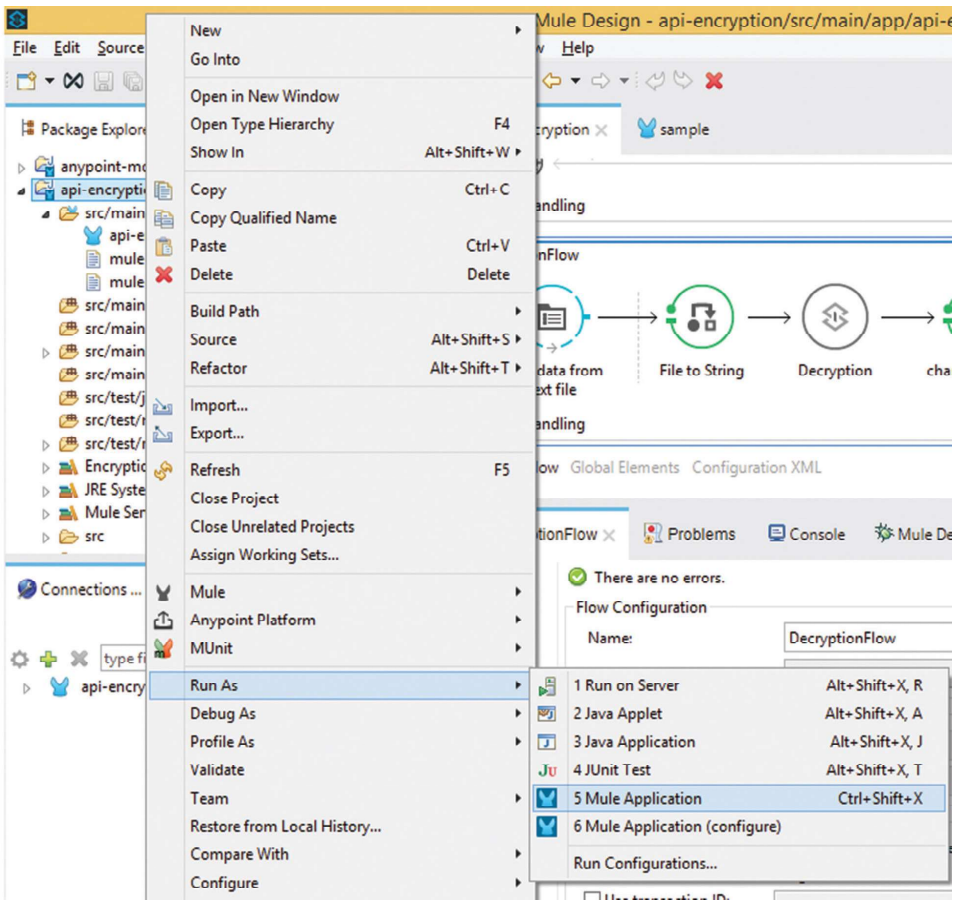

Figure 26.24: Run Application

15. After some time, Application will be started as shown below:

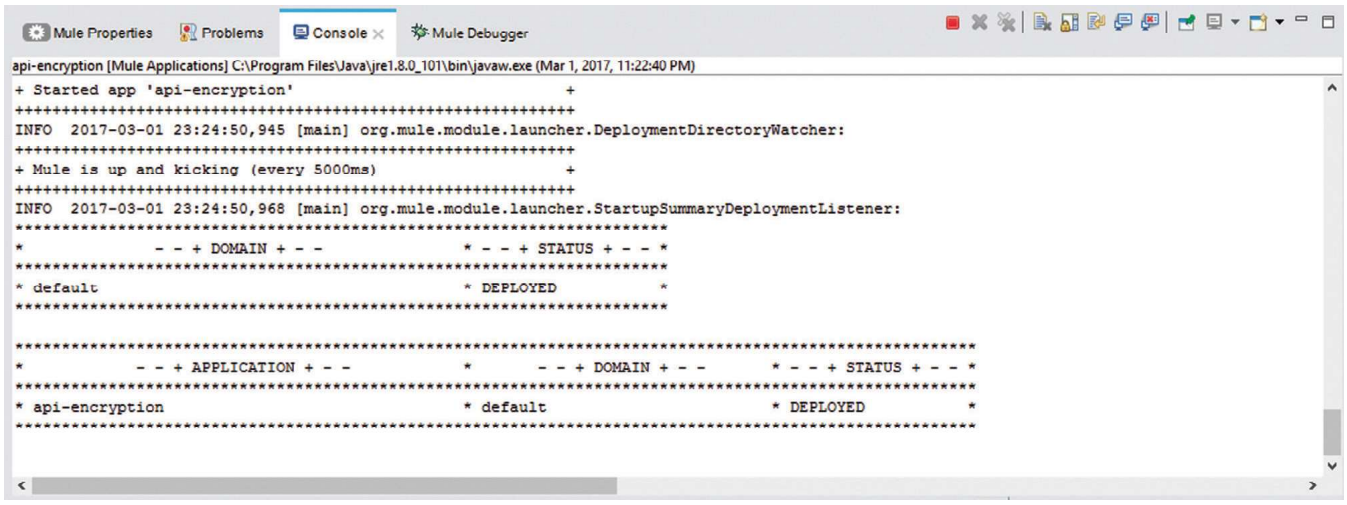

Figure 26.25: Application Started

16. Now, Application is deployed at the local server with URL http://localhost:8081/jceEncryptor.

C. Testing

1. Open the browser and hit the deployed API http://localhost:8081/jceEncryptor. Here, Input is provided as 'kishan'.

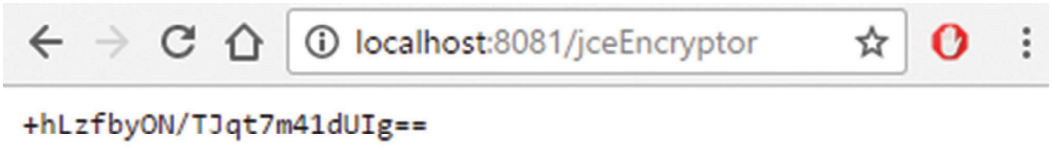

Figure 26.26: Encrypted Result

2. The first flow will encrypt the input 'kishan' to '+hLzfbyON/TJqt7m41dUIg==' and print it into sample.txt file at the path provided.
3. The 2nd flow will come into action once the file is saved in a test folder because the File connector is used as an inbound endpoint.
4. It will decrypt the input to 'kishan' because same key is provided in jce encrypter configuration.
5. After decryption, decrypted data gets implemented and printed to final.txt file at the path provided.
6. Check E:\backup\original path. A file named as final.txt should be available as shown below. Here, the file is opened in Notepad++. However, it can be opened in Notepad or Textpad, etc.

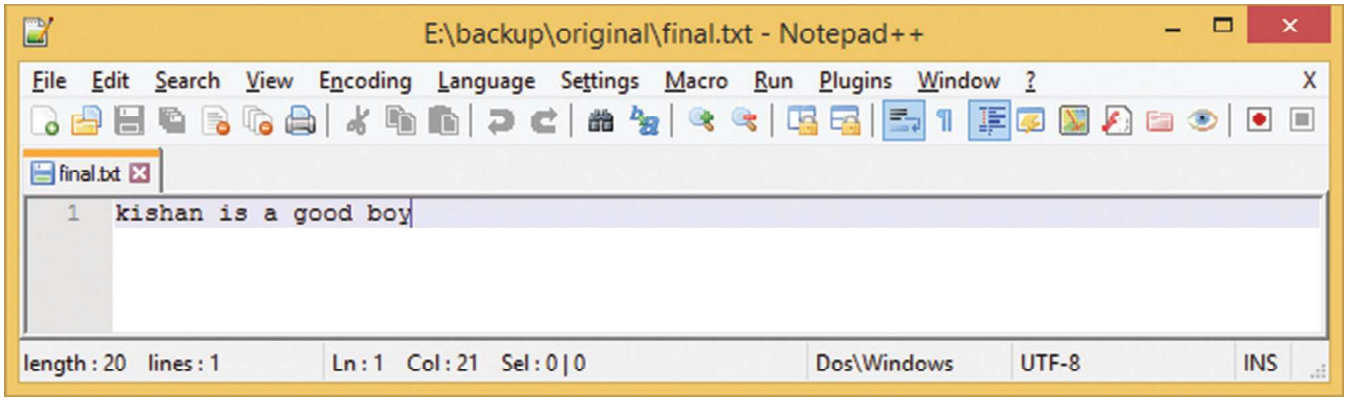

Figure 26.27: Decrypted Result

Anticipated Issues

1. Installation of security provider is required to get Encryption connector in Mule Palette.
2. Folder should be created which is mentioned in pre-requisites.
3. Key should be the same for both encryption and decryption.

References

1. https://docs.mulesoft.com/mule-user-guide/v/3.7/installing-anypoint-enterprise-security
2. http://blog.avenuecode.com/mule-jce-encryption
3. https://docs.mulesoft.com/mule-user-guide/v/3.7/mule-message-encryption-processor