

Oracle Financial Services Crime and Compliance Studio

Administration Guide

Release 8.0.7.3.0

March 2020

E91246.01

ORACLE®
Financial Services

OFS Crime and Compliance Studio

Copyright © 2020 Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are “commercial computer software” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

For information on third party licenses, click [here](#).

Document Control

Version Number	Revision Date	Changes Done
8.0.7.3.0	Updated: March 2020	<ul style="list-style-type: none">• The Financial Crime Graph Model has been modified from Homogenous to Heterogenous format, that is, the node and edge types can have specific properties that are relevant to their type.• The following changes have been introduced to the graph model to make the graph more suitable for users initiating investigations.<ul style="list-style-type: none">■ The intermediaries in the Out-of-the-box graphs have been removed and are not loaded by default for ease of investigation.■ The derived entity node logic has been changed such that the Name and ID information are combined in the derived entity nodes to prevent multiple transaction edges to entities with the same name. <p>For more information, see “Oracle Financial Crime Graph Model” on page 6.</p> <ul style="list-style-type: none">• The ETL process has been changed as there is no longer a need to homogenize the nodes and edges before loading into the graph. For more information, see “Appendix A - Creating and Executing Run Executable” on page 51.
8.0.7.2.0	Updated: February 2020	Updated the guide for 8.0.7.2.0 release.
8.0.7.1.0	Updated: October 2019	Updated the guide for 8.0.7.1.0 release.
8.0.7.0.0	Created: Feb 2019	Created the first version of the Crime and Compliance Studio Administration Guide for 8.0.7.0.0 Release.

Table of Contents

1	About this Guide.....	4
1.1	Summary.....	4
1.2	Audience	4
1.3	Related Documents.....	4
1.4	Abbreviations	4
2	About Oracle Financial Services Crime and Compliance Studio	5
2.1	Introduction to Crime and Compliance Studio	5
2.2	The Architecture of Crime and Compliance Studio	6
2.3	Oracle Financial Crime Graph Model	6
3	Managing User Administration	8
3.1	Managing Identity and Authorization.....	8
3.1.1	<i>Identity and Authorization Process Flow</i>	<i>8</i>
3.2	Granting Permissions	9
4	Accessing Crime and Compliance Studio	10
5	Managing FCC Studio Batches.....	12
5.1	Preparing for Batches.....	12
5.2	Performing Batches	12
5.2.1	<i>Data Movement and Graph Loading for Big Data Environment</i>	<i>12</i>
5.2.2	<i>Executing Published Notebook</i>	<i>14</i>
6	Configuring Security for PGX.....	16
6.1	Prepare Certificates.....	16
6.1.1	<i>Create a Self-Signed Server Certificate</i>	<i>16</i>
6.1.2	<i>Import Existing Certificate or Install Certificate from a Certificate Authority</i>	<i>17</i>
6.2	Prepare Client Keystore	18
6.3	Prepare Client Truststore	18
6.4	Configure PGX Server.....	19
6.5	Test Connection Using PGX Client Shell	19
7	Configuring Interpreters	20
7.1	Accessing Interpreters.....	20
7.2	Creating New Interpreter Variant.....	20

7.2.1	<i>Creating a New JDBC Interpreter Variant</i>	21
7.3	Configure Interpreters.....	21
7.3.1	<i>fcc-jdbc Interpreter</i>	22
7.3.2	<i>fcc-ore Interpreter</i>	23
7.3.3	<i>fcc-pyspark Interpreter</i>	25
7.3.4	<i>fcc-python Interpreter</i>	26
7.3.5	<i>fcc-spark-scala Interpreter</i>	27
7.3.6	<i>fcc-spark-sql Interpreter</i>	28
7.3.7	<i>jdbc</i>	29
7.3.8	<i>md Interpreter</i>	30
7.3.9	<i>pgql Interpreter</i>	30
7.3.10	<i>pgx-algorithm Interpreter</i>	31
7.3.11	<i>pgx-java Interpreter</i>	31
7.3.12	<i>pyspark Interpreter</i>	31
7.3.13	<i>spark Interpreter</i>	32
8	Managing Tasks	34
8.1	Accessing Tasks.....	34
8.2	Task Statuses	34
8.3	Table Columns	35
8.4	Table Filters	35
9	Managing Permissions	37
9.1	Permissions Overview	37
9.2	Accessing Permissions Page.....	39
9.2.1	<i>Users</i>	40
9.2.2	<i>Groups</i>	40
9.2.3	<i>Roles</i>	40
9.2.4	<i>Permission Templates</i>	41
10	Managing Credentials	42
10.1	Accessing Credentials	42
10.2	Using Credentials	42
10.2.1	<i>Link Credentials to Interpreter Variants</i>	42

11	Configuring ETL	45
11.1	Performing Data Source Configuration.....	45
11.2	Performing Graph Configuration.....	48
11.2.1	<i>Attributes Case in Graph</i>	<i>48</i>
11.2.2	<i>Extra Empty Nodes and Edges Providers</i>	<i>49</i>
11.2.3	<i>Additional Configuration</i>	<i>50</i>
12	Appendix A - Creating and Executing Run Executable	51
13	Appendix B - Adding Packages to Python Interpreter	56

1 About this Guide

1.1 Summary

This guide provides instructions for maintaining and configuring the OFS Crime and Compliance Studio (FCC Studio) application, its subsystem components, and any third-party software required for operation.

FCC Studio provides an open and scalable infrastructure that supports end-to-end functionality across all Oracle Financial Services solution sets. FCC Studio's extensible and modular architecture enables a customer to deploy new solution sets readily as the need arises.

1.2 Audience

This guide is intended for administrators and implementation consultants. Their roles and responsibilities, as they operate within FCC Studio, include the following:

- **System Administrator:** Configures and maintains the system, user accounts, and roles, monitors data management, archives data, loads data feeds, reloads cache and performs post-processing tasks.

1.3 Related Documents

You can access the following additional documents related to the OFS Crime and Compliance Studio application from the [Oracle Help Center \(OHC\)](#) Documentation Library:

- *Oracle Financial Services Crime and Compliance Studio Installation Guide*
- *Oracle Financial Services Crime and Compliance Studio Deployment Guide (Using Kubernetes)*
- *Oracle Financial Services Crime and Compliance Studio User Guide*
- *Oracle Financial Services Crime and Compliance Studio Data Model Guides*
- *Oracle Financial Services Crime and Compliance Studio Release Notes and Readme*

1.4 Abbreviations

The following table lists the abbreviations used in this document:

Table 1 Abbreviations

Abbreviation	Meaning
OFS	Oracle Financial Services
FCC Studio	Financial Crime and Compliance Studio
OFSAA	Oracle Financial Services Analytical Application
PGX	Parallel Graph AnalytiX
AML	Anti-money Laundering
BD	Behavior Detection

2 About Oracle Financial Services Crime and Compliance Studio

This chapter provides functional details about Oracle Financial Services (OFS) Crime and Compliance Studio (FCC Studio) application.

This chapter includes the following sections:

- [Introduction to Crime and Compliance Studio](#)
- [The Architecture of Crime and Compliance Studio](#)
- [Oracle Financial Crime Graph Model](#)

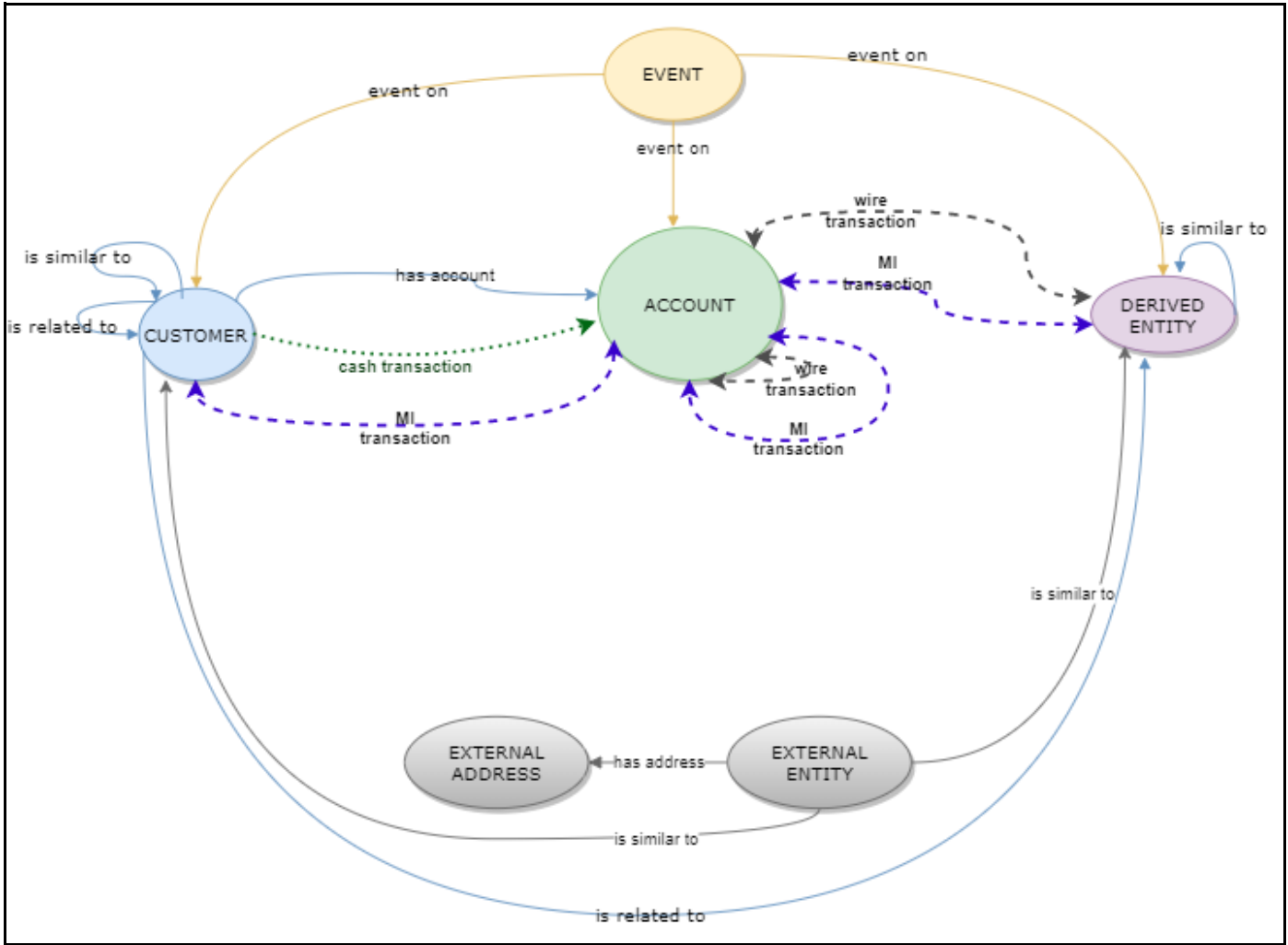
2.1 Introduction to Crime and Compliance Studio

To effectively monitor anti-money laundering and anti-fraud programs in financial institutions, the most challenging requirement is to quickly identify and adapt to the changing patterns of financial crime. This ability to discover new and emerging criminal behavioral patterns, coupled with the facility to rapidly deploy as models, is a critical requirement.

Data scientists and analysts can use FCC Studio to interactively explore financial crime data and gain insights into new and emerging financial crime patterns and trends.

The key features of FCC Studio include the following:

- Provides an integrated and comprehensive analytics toolkit designed to rapidly discover and model new financial crime patterns.
- Interacts with the database, process the data, and generate patterns in various formats using interpreters.
- Provides secure access to an institution's financial crime data with predefined scenarios, out-of-the-box graph queries, and visualizations.
- Uses Graph Analytics and Graph Query methods to analyze historic data available in the database, and forecast the generated patterns using various interpreters.
- Uses Machine Learning Algorithms to gain insights from historical alert data to prioritize the alerts generated by the detection engines.
- Offers a unified tool for Graph Analytics, Data Visualization, Machine Learning, Scenario Authoring, Pattern Discovery, Data Mashups and testing for financial crime data.
- Works with Apache Spark, the most prevalent analytics engine on Big Data.
- Works with Apache Zeppelin, a web-based notebook that enables interactive data analysis.
- Supports Polyglot Scenario Authoring to author new scenarios in SQL, Scala, Python, or R language.
- Embedded with highly scalable in-memory Graph Analytics Engine (PGX).
- Enterprise-ready with underlying OFSAA framework.
- Works with earlier 8.x releases of Oracle Financial Crime and Compliance Management Anti Money Laundering (AML) and Fraud applications.
- Integrated with Oracle Financial Crime Application Data and readily usable across the enterprise financial crime data lake. This can automatically load Oracle AML and Fraud data into the data lake and mashup FCC Studio data with third-party data for discovery and modeling.



3 Managing User Administration

This chapter provides information on creating users who can access FCC Studio and execute batches. Creation of users and execution of batches must be performed in the OFSAA environment.

User administration involves creating and managing users and providing access to FCC Studio based on assigned roles.

This section covers the following topics:

- [Managing Identity and Authorization](#)
- [Granting Permissions](#)

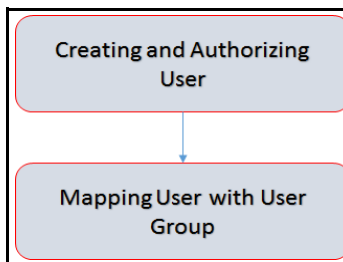
3.1 Managing Identity and Authorization

This section provides information on creating, mapping and authorizing users, and providing access to FCC Studio. It covers the following topics:

- [Identity and Authorization Process Flow](#)
- [Creating and Authorizing User](#)
- [Mapping User with User Group](#)

3.1.1 Identity and Authorization Process Flow

The following figure shows the process flow of identity management and authorization.



The following table lists the various actions involved in the user administration process flow:

Table 1 User Administration Process Flow

Action	Description
Creating and Authorizing User	Create a user by providing the user name, user designation, and time period during which the user will be active in FCC Studio.
Mapping User with User Group	Map user with a user group that provides the user with the privileges defined to that user group.

3.1.1.1 Creating and Authorizing User

The users with SYSADMN and SYSAUTH roles can create and authorize users respectively. For more information on creating and authorizing users, see [Oracle Financial Services Analytical Applications Infrastructure User Guide](#).

3.1.1.2 Mapping User with User Group

A user is mapped with a user group, and the user group is associated with a role. Each role comprises of certain predefined privileges.

On mapping a user to a user group, the user is granted with the privileges that are defined to the role of the user group. The SYSADM user maps a user to a user group in FCC Studio.

The following table describes the roles and the corresponding user groups in FCC Studio.

Table 2 Roles and User Groups in FCC Studio

Role	User Groups
DSADMIN	DSADMINGRP
DSINTER	DSINTERGRP
DSUSER	DSUSERGRP
DSBATCH	DSBATCHGRP

You can log in to the FCC Studio application installed with non-OFSAA as one of the following users:

- DSADMIN
- DSUSER
- DSBATCH

3.2 Granting Permissions

To grant permissions, follow these steps:

1. Log in to Oracle Database from as a SYSDBA user.
2. Execute the following command:

```
grant execute dbms_ols to <Studio_DB_Username>;
```

The Execute permission is granted to VPD.

3. Execute the following command:

```
grant create any context to <STUDIO_DB_USER_NAME>;
```

The Create permission is granted to context.

4 Accessing Crime and Compliance Studio

To access the Crime and Compliance Studio application as a system administrator, follow these steps:

1. Enter the URL in the following format into the browser:

`https://<Host_Name>:<Port_Number>`

Here <Port_Number> is,

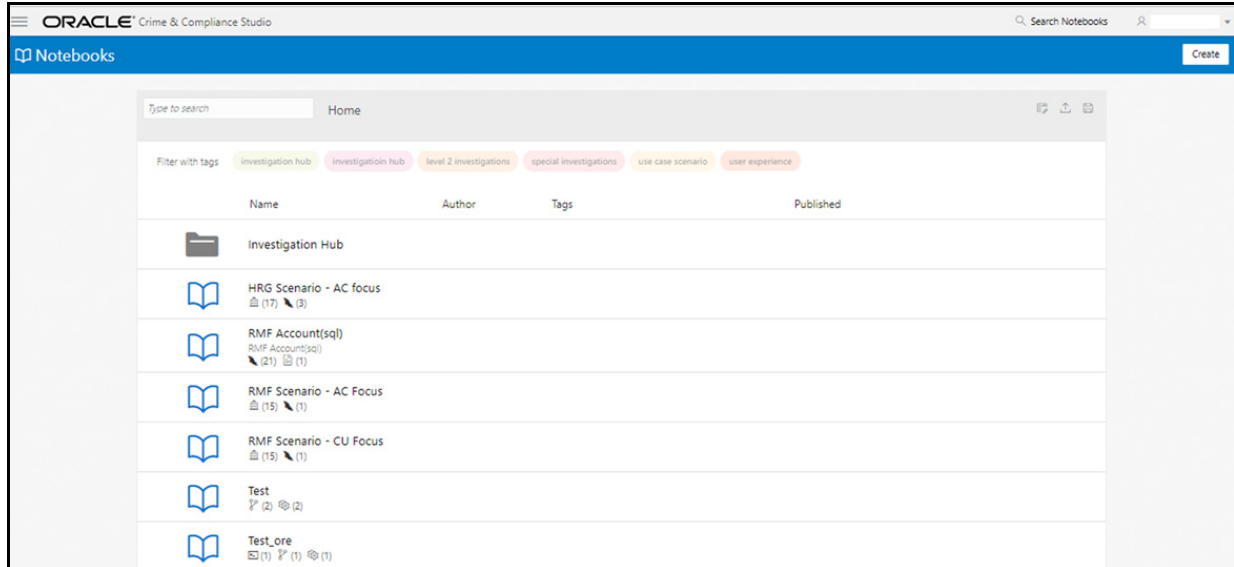
- 7008 for FCC Studio installed on-premise.
- 30078 for FCC Studio deployed on the Kubernetes cluster.


The Crime and Compliance Studio login page is displayed.

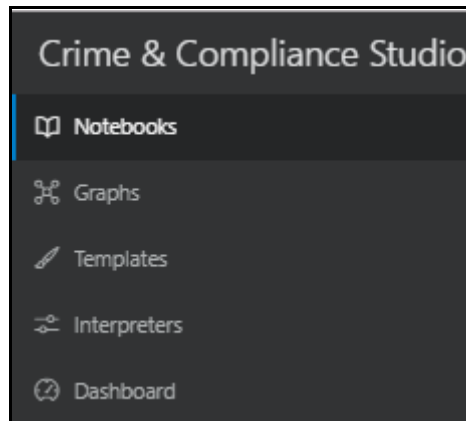


2. Login with the **Username** and **Password** of the System Administrator.
3. Click **Login**.

The Crime and Compliance Studio application's landing page is displayed.



4. Click the  **Menu** icon on the top left corner.
The menu items applicable to the logged-in user are displayed.



For information on Notebooks, Graphs, and Templates, see [Oracle Financial Services Crime and Compliance Studio User Guide](#).

5 Managing FCC Studio Batches

This chapter provides information on creating batches required for FCC Studio. Batches enable us to load graphs, run notebooks, and move data from Oracle Database or Big Data to FCC Studio.

This section covers the following topics:

- [Preparing for Batches](#)
- [Performing Batches](#)

5.1 Preparing for Batches

Follow these steps to prepare the batches:

1. Copy all the jars from the `<STUDIO_INSTALLATION_PATH>/ficdb/lib` directory to the `<FIC_HOME of OFSAA_Installed_Path>/ficdb/lib` directory.
2. Copy the `NBExecutor.txt` file from the `<STUDIO_INSTALLATION_PATH>/ficdb/bin` directory to the `<FIC_HOME of OFSAA_Installed_Path>/ficdb/bin` directory.
3. Navigate to the `<Studio_Installation_Path>/ficdb/bin` directory.
4. Run the `FCCM_Studio_Set_UserPass.sh` command as follows:
 - `FCCM_Studio_Set_UserPass.sh --username "Username" --password "Password"`
 - or
 - `FCCM_Studio_Set_UserPass.sh -u "USERNAME" -p "PASSWORD"`

The `FCC_Studio_SecretKey.properties` and `NBExecutor.txt` files are created in the `<Studio_Installation_Path>/ficdb/conf` directory.

NOTE

1. Ensure that the `FCC_Studio_SecretKey.properties` and `NBExecutor.txt` files are present in the `<Studio_Installation_Path>/ficdb/conf` directory, before executing a notebook batch.
2. If only the `NBExecutor.txt` file is present in the `<Studio_Installation_Path>/ficdb/conf` directory, then re-execute the `FCCM_Studio_Set_UserPass.sh` command with username and password to create a new `FCC_Studio_SecretKey.properties` file and updated `NBExecutor.txt` file.

5.2 Performing Batches

The following type of batches are configured in FCC Studio.

- [Data Movement and Graph Loading for Big Data Environment](#)
- [Executing Published Notebook](#)

5.2.1 Data Movement and Graph Loading for Big Data Environment

You can move data from the *BD Atomic* environment to the *Big Data* environment. As a result, a graph is created in the form of the `.CSV` and `config.json` files. These files can be used to view and query

the PGQL and PGX graphs. For information on the Oracle Financial Crime Graph Model, see [Oracle Financial Crime Graph Model](#).

1. To create batches for data movement and graph loading, perform the following:
 - For FCC Studio with OFSAA: [Appendix A - Creating and Executing Run Executable](#)
 - For FCC Studio with non-OFSAA: [Executing Sqoop Job](#)
2. Verify indices in elastic search. For more information, see [Verifying Indices in Elastic Search](#).

5.2.1.1 Executing Sqoop Job

NOTE This section is applicable for FCC Studio with non-OFSAA.

To execute the Sqoop job for data movement and graph loading for FCC Studio without OFSAA, follow these steps:

1. Execute the `FCCM_Studio_SqoopJob.sh` command with the required parameters as follows:

```
./FCCM_Studio_SqoopJob.sh <Batch_Name> <Batch_ID> EXEC <FROM_FIC_MIS_-  
DATE> <TO_FIC_MIS_DATE> SNAPSHOT_DT=<SNAPSHOT_DATE>, DATAMOVEMENTCODE=ALL  
<SOURCE FOR DATAMOVEMENT> <SOURCE TO INCLUDE IN GRAPH>
```

For example:

```
./FCCM_Studio_SqoopJob.sh A B C 20150618 20190618 SNAP-  
SHOT_DT=20181219, DATAMOVEMENTCODE=ALL FCDM FCDM, ICIJ
```

Where,

- <Batch_Name> is A
- <Batch_ID> is B
- EXEC is C
- <FROM_FIC_MIS_DATE> is 20150618
- <TO_FIC_MIS_DATE> is 20190618
- <SNAPSHOT_DATE> is 20181219
- DATAMOVEMENTCODE is ALL
- <SOURCE FOR DATAMOVEMENT> is FCDM
- <SOURCE TO INCLUDE IN GRAPH> is FCDM, ICIJ.

Ensure that the values are comma separated.

5.2.1.2 Verifying Indices in Elastic Search

To verify indices in elastic search, follow these steps:

1. Enter the URL in the following format into the browser:

```
http://<Elastic_Search_Hostname>:<Elastic_Search_Port>/_cat/indices
```

All the indices must be displayed with the same snapshot date with which the job is triggered with.

Format: <Index name>_<Snapshot Date>

For example:

- fcdm_customer_2020-03-01
- icij_bahama_external_address_2020-03-01

5.2.2 Executing Published Notebook

The published notebook can be scheduled for execution with a set of threshold values required for generating alerts or trends.

To execute published notebook, perform the following:

- For published scenario notebook:
 - For FCC Studio with OFSAA, [Appendix A - Creating and Executing Run Executable](#)
 - For FCC Studio with non-OFSAA, [Executing Published Scenario Notebook for FCC Studio with Non-OFSAA](#)
- For published non-scenario notebook:
 - For FCC Studio with OFSAA, [Appendix A - Creating and Executing Run Executable](#)
 - For FCC Studio with non-OFSAA, [Executing Published Non-Scenario Notebook for FCC Studio with Non-OFSAA](#)

5.2.2.1 Executing Published Scenario Notebook for FCC Studio with Non-OFSAA

NOTE

- This section is applicable for FCC Studio with non-OFSAA.
- Ensure that the username and password are set before executing a notebook batch. For more information, see [Preparing for Batches](#).

Execute the following command with the required parameters as follows:

```
./FCCM_Studio_NotebookExecution.sh "notebookID" "null" "scenarioID" "thresholdsetID" "null" "BATCH_ID"
```

5.2.2.2 Executing Published Non-Scenario Notebook for FCC Studio with Non-OFSAA

NOTE

This section is applicable for FCC Studio with non-OFSAA.

To execute a non-scenario notebook for FCC Studio with non-ofsaa, follow these steps:

1. Execute the following command:

```
./FCCM_Studio_NotebookExecution.sh "notebookID" "null" "null" "null" "paramkey1~~value1@@paramkey2~~value2"
```

For example, If

- paramkey1 is ficmisdate
- paramkey2 is lookbackperiod
- value1 is 20-09-2018
- value2 is 30

Then the extraparams must be written as follows:

ficmisdate~~20-09-2018@@lookbackperiod~~30

6 Configuring Security for PGX

The PGX web server enables two-way SSL/TLS by default. The PGX server enforces TLS 1.2 and disables certain cipher suites known to be vulnerable to attacks. Upon TLS handshake, both server and client present certificates to each other which are used to validate the authenticity of the other party. Client certificates are additionally used to authorize client applications.

This chapter includes the following sections.

- [Prepare Certificates](#)
- [Prepare Client Keystore](#)
- [Prepare Client Truststore](#)
- [Configure PGX Server](#)
- [Test Connection Using PGX Client Shell](#)

6.1 Prepare Certificates

NOTE Disabling SSL/TLS:
You can skip this part if you turn off SSL/TLS in a single-node or multi-node PGX server configuration. However, we strongly recommend leaving SSL/TLS turned on for any production deployment.

You must create a server certificate that will be validated by the client upon SSL/TLS handshake. You can either create a self-signed server certificate or import a certificate from a certificate authority.

This section includes the following:

- [Create a Self-Signed Server Certificate](#)
- [Import Existing Certificate or Install Certificate from a Certificate Authority](#)

6.1.1 Create a Self-Signed Server Certificate

NOTE Do not use self-signed certificates in production deployments. For production, you should obtain a certificate from a certificate authority that is trusted by your organization.

You can create a self-signed certificate to the keytool command-line utility, which is part of the Java Development Kit (JDK) that you already installed.

Perform the following to create a self-signed server certificate:

- [Create New Keystore](#)
- [Extract the Certificate](#)

6.1.1.1 Create New Keystore

Perform the following to create a new keystore.

1. Create a new keystore containing a self-signed certificate by executing the following command:

```
keytool -genkey -alias pgx -keyalg RSA -keystore server_keystore.jks
```

The command prompts for keystore password, general information of the certificate (which will be displayed to clients who attempt to connect to the PGX web server) and the key password. The keystore password is for the keystore file itself and the key password is for the certificate. This is because JKS keystore files can store more than one certificate (identified by the provided alias).

2. Upon prompt, enter the first name, last name, and hostname of the host you will deploy the PGX server on.

NOTE

- If the hostname in the certificate does not match the hostname the server is listening on, client applications will reject the connection.
- In distributed mode: use the host which starts the web-server

For the multi-node setup, use the first hostname in the list of `pgx_hostnames` as the hostname for your certificates. Only the first host in this list will start an http server.

6.1.1.2 Extract the Certificate

The PGX server requires both the server certificate and server private key in the PKCS12 (PEM) format.

Perform the following to extract the certificate and private key from the JKS file:

1. Convert the generated `server_keystore.jks` file into a `server_keystore.p12` file by executing the following:

```
keytool -importkeystore -srckeystore server_keystore.jks -destkeystore
server_keystore.p12 -srcalias pgx \
    -srcstoretype jks -deststoretype pkcs12
```

The command will prompt with both the source and destination keystore password.

2. Enter the source and destination keystore password.

A file `server_keystore.p12` is generated in the current directory.

3. Extract certificate and private key from that `server_keystore.p12` file by executing the following openssl commands:

```
openssl pkcs12 -in server_keystore.p12 -nokeys -out server_cert.pem
openssl pkcs12 -in server_keystore.p12 -nodes -nocerts -out
server_key.pem
```

The `server_cert.pem` and `server_key.pem` are generated in the current directory.

6.1.2 Import Existing Certificate or Install Certificate from a Certificate Authority

Refer [Tomcat TLS/SSL documentation](#) on how to import existing certificates or on how to install a certificate from a certificate authority into keystore files.

6.2 Prepare Client Keystore

NOTE Disabling two-way SSL/TLS

You can skip this part if you turn off client authentication in single-node or multi-node PGX server configuration. However, we strongly recommend leaving two-way SSL/TLS turned on for any production deployment.

For two-way SSL/TLS to work, you have to create one certificate for each client application you allow access to your PGX server. You must first create a keystore file for the client.

1. Execute the following to create a keystore file for the client:

```
keytool -genkey -alias pgx -keyalg RSA -keystore client_keystore.jks
```

The above command prompts with a keystore password, general information of the certificate and the key password. Note down the general information in the certificate (distinguished name string) as you will need this information in the next section for the PGX server authorization configuration.

2. You must sign the certificate inside the client keystore with the server private key which will make the client certificate to be accepted by the server. For which you must first create a sign request file `client.csr` by executing the following:

```
keytool -certreq -keystore client_keystore.jks -storepass <keystore_-\npassword> -alias pgx -keyalg RSA -file client.csr
```

3. Sign the `client.csr` file by providing both the server's certificate and private key files to the following openssl command:

```
openssl x509 -req -CA server_cert.pem -CAkey server_key.pem -in cli-\nent.csr -out client_certificate.pem -days 365 -CAcreateserial
```

A signed client certificate file `client_certificate.pem` is generated that is accepted by the server for the next 365 days. You can modify the `-days` parameter as per your needs.

4. Import both the server certificate as well as the signed client certificate back into the client keystore file by executing the following:

```
keytool -import -noprompt -trustcacerts -keystore client_keystore.jks -file\nserver_cert.pem -alias pgxserver
```

```
keytool -import -noprompt -trustcacerts -keystore client_keystore.jks -file\nclient_certificate.pem -alias pgx
```

6.3 Prepare Client Truststore

Use the same `client_keystore.jks` file for both the client keystore (which certificate to present to the server) and the client trust store (which server certificates to trust). If you have used a self-signed server certificate, you also have to import the server certificate's trust authority (CA) into the client keystore, else the client will reject the server certificate. Note that if you are using the PGX client shell, a range of well-known certificate authorities are trusted already by default by the client-side Java virtual machine.

6.4 Configure PGX Server

Specify the paths to the `server_cert.pem` and the `server_key.pem` files in the single-node or multi-node PGX server configurations. You can also specify a list of certificate authorities that will be trusted by the server.

6.5 Test Connection Using PGX Client Shell

If you started the webserver with a self-signed certificate, you must first configure the client to accept the certificate. As the certificate is self-signed and not issued from a trusted certificate authority, the PGX client would reject it otherwise. You can set the trust store of the PGX client shell via the `--truststore` command-line option. Similarly, we have to specify the path to the keystore (`--keystore`) which contains the certificate the client will present to the server for authentication and authorization as well as the keystore password (`--password`).

NOTE

Do not accept self-signed certificates from unknown sources. Do not accept certificates from sources other than yourself.

Assuming the PGX web server listens on the default port 7007 on the same machine and you created the keystores as described above, you can test the connection by executing the following:

```
cd $PGX_HOME  
  
./bin/pgx --base_url https://localhost:7007 --truststore client_keystore.jks  
--keystore client_keystore.jks --password <keystore_password>
```

If the shell starts up without any error, you successfully connected to the PGX web server securely over two-way TLS/SSL.

7 Configuring Interpreters


An interpreter reads and executes the instructions written in a programming or scripting language without compiling the high-level language code into a machine language program.

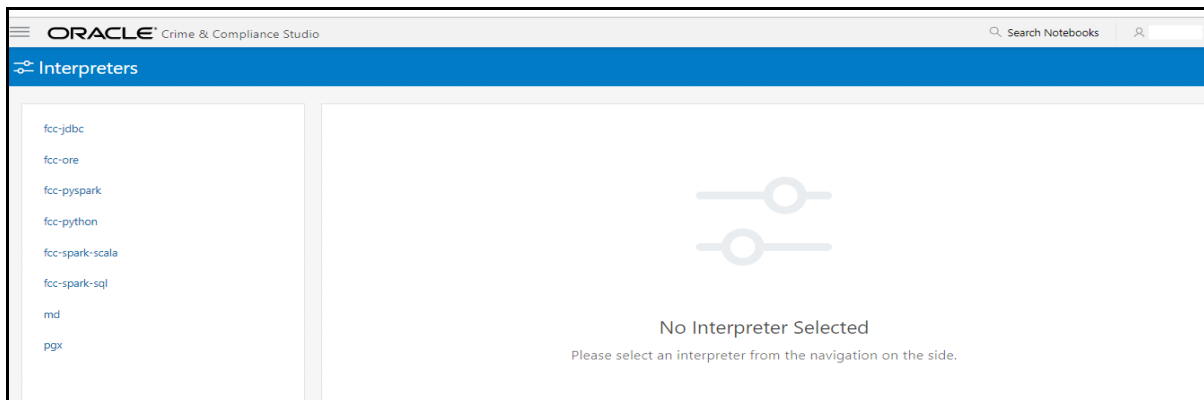
This section covers the following topics:

- [Accessing Interpreters](#)
- [Creating New Interpreter Variant](#)
- [Configure Interpreters](#)

7.1 Accessing Interpreters

To access interpreters, follow these steps:

1. Click the  Menu icon on the upper-left corner on the FCC Studio landing page.
The menu items are listed.
2. Click **Interpreters**.
The *Interpreters* page is displayed.



3. Click the interpreter that you want to access from the list displayed on the LHS.
The default interpreter variant configured is displayed on the RHS.
4. Modify the required values.
5. Click **Update**.
The modified values are updated in the interpreter.

7.2 Creating New Interpreter Variant

In FCC Studio, you can either use a default interpreter variant or create a new variant for an interpreter. You can create more than one variant for an interpreter.

- For a sample on creating a new interpreter variant, see [Creating a New JDBC Interpreter Variant](#)
- To enable a second Spark/PySpark interpreter, see [Enabling a Second Spark/PySpark Interpreter](#) chapter in the [OFS Crime and Compliance Studio Installation Guide \(On-Premise\)](#).

7.2.1 Creating a New JDBC Interpreter Variant

To create a new JDBC interpreter variant:

1. Navigate to the *Interpreters* page.
2. Click the **fcc-jdbc** interpreter from the list displayed on the LHS.
The default interpreter variant is displayed on the RHS.
3. Click the following icon to create a new variant for the selected interpreter:



The **Create Interpreter Variant** dialog box is displayed.

4. Enter the **Name** for the new interpreter variant.
5. Click **Create**.
A new variant is created with name, **<Interpreter Type>.<Variant Name>**.
6. Provide the new schema details such as the **default.url**, **default.user**, and **default.password**.
7. Click **Update**.
A new variant is created for the jdbc interpreter.
8. The Oracle Database schema that you have created must be granted with the same permissions that are granted to the BD atomic schema.

For more information, see the *Prerequisite Environmental Settings* section in the [OFS Crime and Compliance Studio Installation Guide](#).

9. Run the following script after modifying the schema name with the newly created schema:

```
../OFS_FCCM_STUDIO/metaservice/model/SQLScripts/Atomic_Schema/FCC_JRS-DCN_CONTEXT_ATOMIC.sql

../OFS_FCCM_STUDIO/metaservice/model/SQLScripts/Atomic_Schema/PKG_FCC_STUDIO_JURN_VPD.sql

../OFS_FCCM_STUDIO/metaservice/model/SQLScripts/Atomic_Schema/PKG_FCC_STUDIO_JURN_VPD_BODY_ATOMIC.sql
```

10. For using the new interpreter variant in the notebook paragraphs, use the following format:

```
%fcc-jdbc.newVariant
<Your SQL query>
```

11. Configure the required values for the properties.
12. Click **Update**.

A new variant is created for the JDBC interpreter.

7.3 Configure Interpreters

The list of interpreters in FCC Studio are as follows:

- [fcc-jdbc Interpreter](#)

- [fcc-ore Interpreter](#)
- [fcc-pyspark Interpreter](#)
- [fcc-python Interpreter](#)
- [fcc-spark-scala Interpreter](#)
- [fcc-spark-sql Interpreter](#)
- [jdbc](#)
- [md Interpreter](#)
- [pgql Interpreter](#)
- [pgx-algorithm Interpreter](#)
- [pgx-java Interpreter](#)
- [pyspark Interpreter](#)
- [spark Interpreter](#)

7.3.1 fcc-jdbc Interpreter

The configurations for the ofsaa-jdbc interpreter are given as follows:

Table 1 fcc-jdbc Interpreter

Field	Description
pgx.baseUrl	Enter the pgx.baseUrl URL in this field. This is the location where the data is pushed. For example: <code>http://<HOSTNAME>:7007</code>
default.url	Enter the ofsaa jdbc URL in this field. For example: <code>jdbc:mysql://localhost:5554/world</code>
zeppelin.jdbc.principal	Enter the principal name to load from the keytab.
default.driver	Enter the default ofsaa JDBC driver name. For example: <code>com.mysql.jdbc.Driver</code>
default.completer.ttlInSeconds	Enter the time to live sql completer in seconds.
default.password	Enter the default password.
default.splitQueries	This field indicates the presence of default split queries. Enter “true” or “false”.
default.completer.schemaFilters	Enter comma separated schema filters to get metadata for completions.
ofsaa.session.service.url	Enter the session service URL in this field. For example: <code>http://<HOSTNAME>:7047/session-service</code> Here, <HOSTNAME> refers to the server name or IP where fcc-studio will be installed.

Table 1 fcc-jdbc Interpreter

Field	Description
default.user	Enter the name of the default user in this field. For example: root
zeppelin.jdbc.concurrent.max_connection	Enter the number of maximum connections allowed.
ofsaa.metaservice.url	Enter the metaservice URL in this field. For example: <code>http://<HOSTNAME>:7045/metaservice</code> Here, <HOSTNAME> refers to the server name or IP where fcc-studio will be installed.
common.max_count	Enter the maximum number of SQL result to display.
zeppelin.jdbc.auth.type	Enter the default jdbc authentication type.
zeppelin.jdbc.precode	Enter the snippet of code that executes after the initialization of the interpreter.
zeppelin.jdbc.concurrent.use	Enter to enable or disable concurrent use of JDBC connections. Enter "true" or "false".
zeppelin.jdbc.keytab.location	Enter the keytab location.

7.3.2 fcc-ore Interpreter

The configurations for the fcc-ore interpreter are given as follows:

Table 2 fcc-ore Interpreter

Field	Description
ore.sid	Enter the SID of DB server where the fcc-ore interpreter wants to connect.
rendering.row.limit	Indicates the number of rows to be shown in the fcc-ore interpreter output. For example: 1000
ore.conn_string	Enter the DB connection URL with which the fcc-ore interpreter can make the connection to the schema. This field can be left blank.
https_proxy	Enter the Proxy server using which connection to the internet can be established. For example: <code>www-proxy-hqdc.us.oracle.com:80</code>
ore.type	Enter the fcc-ore interpreter type as Oracle.
ore.password	Enter the schema password where the fcc-ore interpreter wants to connect.

Table 2 fcc-ore Interpreter

Field	Description
libpath	Indicates the custom library path from where R packages will be installed via FCC Studio and will be added to R lib Path. Enter the path to be mentioned under the home directory where FCC Studio is installed. For example: If you want the packages to be available under <code>/home/user/library</code> , and FCC Studio is installed at <code>/home/user/data-studio</code> , then mention <code>/library</code> as the libpath.
ore.host	Enter the hostname of the DB server where the fcc-ore interpreter wants to connect.
rserve.password	Indicates the Rserve password.
rendering.numeric.format	Indicates the Number of digits to round off. For example: <code>%.2f</code>
ore.service_name	Enter the Service Name of DB server where the fcc-ore interpreter wants to connect.
rserve.try.wrap	Enter False.
rserve.host	Indicates the Rserve host.
repo_cran	Indicates the CRAN URL from where R libraries are downloaded to install R packages. For example: <code>https://cran.r-project.org/</code>
ofsa.sessionservice.url	Enter the session service URL in this field. For example: <code>http://<HOSTNAME>:7047/sessionservice</code> Here, <code><HOSTNAM></code> refers to the server name or IP where fcc-studio will be installed.
ore.all	Indicates all tables are synced to the fcc-ore interpreter. Enter the value as True.
rserve.plain.qap.disabled	Indicates whether plain QAP is disabled in the server or not. If disabled, the connection will always be attempted using SSL. For example: False
ore.user	Enter the schema name where the fcc-ore interpreter wants to connect.
http_proxy	Enter the Proxy server using which connection to the internet is established. This value is used to set the initial setting that makes the environment compatible to download the libraries available in R. For example: <code>www-proxy-hqdc.us.oracle.com:80</code>
rserve.port	Indicates the Rserve port.
rserve.secure.login	Enter TRUE to enforce secure login.

Table 2 fcc-ore Interpreter

Field	Description
rendering.knitr.options	Enter the Knitr output rendering option. For example: <code>out.format = 'html', comment = NA, echo = FALSE, results = 'verbatim', message = F, warning = F, dpi = 300</code>
rserve.user	Indicates the Rserve username.
ore.port	Enter the port number of the DB server where the fcc-ore interpreter wants to connect.
ofsaaservice.url	Enter the metaservice URL in this field. For example: <code>http://<HOSTNAME>:7045/metaservice</code> Here, <HOSTNAME> refers to the server name or IP where fcc-studio will be installed.
rendering.include.row.name	Indicates whether to include row names. For example: <code>false</code>
rendering.knitr.image.width	Indicates the image width specification for ore output. For example: <code>60</code>

7.3.3 fcc-pyspark Interpreter

The configurations for the fcc-pyspark interpreter are given as follows:

Table 3 fcc-pyspark Interpreter

Field	Interpreter
pgx.baseUrl	Enter the pgx.baseUrl URL in this field. This is the location where the data is pushed. For example: <code>http://##HOSTNAME##:7007</code>
livy.spark.executor.instances	Enter the number of executors to launch for the current session.
livy.spark.dynamicAllocation.cachedExecutorIdleTimeout	Enter the cached execution timeout in seconds.
zeppelin.livy.url	Enter the Livy URL in this field. Livy is an interface between Data Studio and Spark. For example: <code>http://##HOSTNAME##:8998</code>
zeppelin.livy.pull_status.interval.millis	Enter the data pull interval in milliseconds.
livy.spark.executor.memory	Enter the amount of memory to use for the executor process.
livy.spark.dynamicAllocation.enabled	This field indicates whether Dynamic Allocation is enabled or not. Enter "true" or "false".
livy.spark.dynamicAllocation.minExecutors	Enter the minimum number of required Dynamic Allocation executors.

Table 3 fcc-pyspark Interpreter

Field	Interpreter
livy.spark.executor.cores	Enter the number of executor cores to use for the driver process.
zeppelin.livy.session.create_timeout	Enter the Zeppelin session creation timeout in seconds.
zeppelin.livy.spark.sql.max-Result	Enter the maximum number of results that need to be fetched.
livy.spark.jars.packages	Enter to add extra libraries to a livy interpreter.
livy.spark.driver.cores	Enter the number of driver cores to use for the driver process.
zeppelin.livy.displayAppInfo	This field indicates whether the application information needs to be displayed or not. Enter “true” or “false”.
livy.spark.driver.memory	Enter the amount of memory to use for the driver process.
zeppelin.livy.principal	Enter the principal name to load from the keytab.
ofsa.sessionservice.url	Enter the session service URL in this field. For example: http://##HOSTNAME##:7047/sessionservice Here, ##HOSTNAME## refers to the server name or IP where fcc-studio will be installed.
ofsa.metaservice.url	Enter the metaservice URL in this field. For example: http://##HOSTNAME##:7045/metaservice Here, ##HOSTNAME## refers to the server name or IP where fcc-studio will be installed.
zeppelin.livy.keytab	Enter the keytab location.
livy.spark.dynamicAllocation.maxExecutors	Enter the maximum number of required Dynamic Allocation executors.

7.3.4 fcc-python Interpreter

The configuration for the fcc-python interpreter are given as follows:

Table 4 fcc-python Interpreter

Field	Description
zeppelin.python	Enter the Python installed path. The value points to the default Python version set for the interpreter. NOTE: To use a different version of Python, you can update the value or you can create an interpreter variant with the new value and point it to the newly created Python version. For example: To use python 3.6, you can create a new fcc-python interpreter variant and enter the value as python 3.6
zeppelin.python.useIPython	Set to true to use the IPython, else set to false.
zeppelin.python.maxResult	Enter the maximum number of results that need to be fetched.

To add desired packages to fcc-python interpreter, see [Chapter 13, "Appendix B - Adding Packages to Python Interpreter"](#).

7.3.5 fcc-spark-scala Interpreter

The configurations for the fcc-spark-scala interpreter are given as follows:

Table 5 fcc-spark-scala Interpreter

Field	Description
pgx.baseUrl	Enter the pgx.baseUrl URL in this field. This is the location where the data is pushed. For example: <code>http://<HOSTNAME>:7007</code>
livy.spark.executor.instances	Enter the number of executors to launch for the current session.
livy.spark.dynamicAllocation.cachedExecutorIdleTimeout	Enter the cached execution timeout in seconds.
zeppelin.livy.url	Enter the Livy URL in this field. Livy is an interface between Data Studio and Spark. For example: <code>http://<HOSTNAME>:8998</code>
zeppelin.livy.pull_status.interval.millis	Enter the data pull interval in milliseconds.
livy.spark.executor.memory	Enter the amount of memory to use for the executor process.
livy.spark.dynamicAllocation.enabled	This field indicates whether Dynamic Allocation is enabled or not. Enter "true" or "false".
livy.spark.dynamicAllocation.minExecutors	Enter the minimum number of required Dynamic Allocation executors.
livy.spark.executor.cores	Enter the number of executor cores to use for the driver process.
zeppelin.livy.session.create_timeout	Enter the Zeppelin session creation timeout in seconds.
zeppelin.livy.spark.sql.maxResult	Enter the maximum number of results that need to be fetched.
livy.spark.jars.packages	Enter to add extra libraries to a livy interpreter.
livy.spark.driver.cores	Enter the number of driver cores to use for the driver process.
zeppelin.livy.displayAppInfo	This field indicates whether the application information needs to be displayed or not. Enter "true" or "false".
livy.spark.driver.memory	Enter the amount of memory to use for the driver process.
zeppelin.livy.principal	Enter the principal name to load from the keytab.
ofsa.sessionservice.url	Enter the session service URL in this field. For example: <code>http://<HOSTNAME>:7047/sessionservice</code> Here, <HOSTNAME> refers to the server name or IP where fcc-studio will be installed.

Table 5 fcc-spark-scala Interpreter

Field	Description
ofsaa.metaservice.url	Enter the metaservice URL in this field. For example: <code>http://<HOSTNAME>:7045/metaservice</code> Here, <HOSTNAME> refers to the server name or IP where fcc-studio will be installed.
zeppelin.livy.keytab	Enter the keytab location.
livy.spark.dynamicAllocation.maxExecutors	Enter the maximum number of required Dynamic Allocation executors.
livy.spark.dynamicAllocation.initialExecutors	Enter the initial Dynamic Allocation executors.

7.3.6 fcc-spark-sql Interpreter

The configurations for the fcc-spark-sql interpreter are given as follows:

Table 6 fcc-spark-sql Interpreter

Field	Description
pgx.baseUrl	Enter the pgx.baseUrl URL in this field. This is the location where the data is pushed. For example: <code>http://<HOSTNAME>:7007</code>
livy.spark.executor.instances	Enter the number of executors to launch for the current session.
livy.spark.dynamicAllocation.cachedExecutorIdleTimeout	Enter the cached execution timeout in seconds.
zeppelin.livy.url	Enter the Livy URL in this field. Livy is an interface between Data Studio and Spark. For example: <code>http://<HOSTNAME>:8998</code>
zeppelin.livy.pull_status.interval.millis	Enter the data pull interval in milliseconds.
livy.spark.executor.memory	Enter the amount of memory to use for the executor process.
livy.spark.dynamicAllocation.enabled	This field indicates whether Dynamic Allocation is enabled or not. Enter "true" or "false".
livy.spark.dynamicAllocation.minExecutors	Enter the minimum number of required Dynamic Allocation executors.
livy.spark.executor.cores	Enter the number of executor cores to use for the driver process.
zeppelin.livy.session.create_timeout	Enter the Zeppelin session creation timeout in seconds.
zeppelin.livy.spark.sql.maxResult	Enter the maximum number of results that need to be fetched.
zeppelin.livy.spark.sql.field.truncate	Indicates to truncate field values longer than 20 characters or not. Enter "true" or "false".

Table 6 fcc-spark-sql Interpreter

Field	Description
livy.spark.jars.packages	Enter to add extra libraries to a livy interpreter.
livy.spark.driver.cores	Enter the number of driver cores to use for the driver process.
zeppelin.livy.displayAppInfo	This field indicates whether the application information needs to be displayed or not. Enter “true” or “false”.
livy.spark.driver.memory	Enter the amount of memory to use for the driver process.
zeppelin.livy.principal	Enter the principal name to lead from the keytab.
ofsaa.sessionservice.url	Enter the session service URL in this field. For example: <code>http://<HOSTNAME>:7047/sessionservice</code> Here, <HOSTNAME> refers to the server name or IP where fcc-studio will be installed.
ofsaa.metaservice.url	Enter the metaservice URL in this field. For example: <code>http://<HOSTNAME>:7045/metaservice</code> Here, <HOSTNAME> refers to the server name or IP where fcc-studio will be installed.
zeppelin.livy.keytab	Enter the keytab location.
livy.spark.dynamicAllocation.maxExecutors	Enter the maximum number of required Dynamic Allocation executors.

7.3.7 jdbc

The configurations for the jdbc interpreter are given as follows:

Table 7 jdbc Interpreter

Field	Description
pgx.baseUrl	Enter the pgx.baseUrl URL in this field. This is the location where the data is pushed. For example: <code>http://<HOSTNAME>:7007</code>
default.url	Enter the jdbc URL in this field.
zeppelin.jdbc.principal	Enter the principal name to load from the keytab.
default.driver	Enter the default JDBC driver name.
default.completer.ttlInSeconds	Enter the time to live sql completer in seconds.
default.password	Enter the default password.
default.splitQueries	This field indicates the presence of default split queries. Enter “true” or “false”.
default.completer.schemaFilters	Enter comma separated schema filters to get metadata for completions.

Table 7 jdbc Interpreter

Field	Description
ofsa.sessionservice.url	Enter the session service URL in this field. For example: <code>http://<HOSTNAME>:7047/sessionservice</code> Here, <HOSTNAME> refers to the server name or IP where fcc-studio will be installed.
default.user	Enter the name of the default user in this field.
zeppelin.jdbc.concurrent.max_connection	Enter the number of maximum connections allowed.
ofsa.metaservice.url	Enter the metaservice URL in this field. For example: <code>http://<HOSTNAME>:7045/metaservice</code> Here, <HOSTNAME> refers to the server name or IP where fcc-studio will be installed.
common.max_count	Enter the maximum number of SQL result to display.
zeppelin.jdbc.auth.type	Enter the default jdbc authentication type.
zeppelin.jdbc.precode	Enter the snippet of code that executes after the initialization of the interpreter.
zeppelin.jdbc.concurrent.use	Enter to enable or disable concurrent use of JDBC connections. Enter "true" or "false".
zeppelin.jdbc.keytab.location	Enter the keytab location.

7.3.8 md Interpreter

The configurations for the md interpreter are given as follows:

Table 8 md Interpreter

Field	Description
markdown.parser.type	Enter the markdown parser type.

7.3.9 pgql Interpreter

The configurations for the pgql interpreter are given as follows:

Table 9 pgql Interpreter

Field	Description
graphviz.formatter.class	For example: <code>oracle.datastudio.graphviz.formatter.DataStudioFormatter</code>
graphviz.driver.class	For example: <code>oracle.pgx.graphviz.driver.PgxDriver</code>

Table 9 pgql Interpreter

Field	Description
base_url	Enter the base URL in this field. This is the location where the data is pushed. For example: <code>http://<HOSTNAME>:7007</code>
zeppelin.interpreter.output.limit	For example: 102,400

7.3.10 pgx-algorithm Interpreter

The configurations for the pgx-algorithm interpreter are given as follows:

Table 10 pgx-algorithm Interpreter

Field	Description
graphviz.formatter.class	For example: <code>oracle.datastudio.graphviz.formatter.DataStudioFormatter</code>
graphviz.driver.class	For example: <code>oracle.pgx.graphviz.driver.PgxDriver</code>
base_url	Enter the base URL in this field. This is the location where the data is pushed.

7.3.11 pgx-java Interpreter

The configurations for the pgx-java interpreter are given as follows:

Table 11 pgx-java Interpreter

Field	Description
graphviz.formatter.class	For example: <code>oracle.datastudio.graphviz.formatter.DataStudioFormatter</code>
graphviz.driver.class	For example: <code>oracle.pgx.graphviz.driver.PgxDriver</code>
base_url	Enter the base URL in this field. This is the location where the data is pushed.
zeppelin.interpreter.output.limit	For example: 102,400

7.3.12 pyspark Interpreter

The configurations for the pgx-java interpreter are given as follows:

Table 12 pyspark Interpreter

Field	Description
zeppelin.pyspark.python	For example: python
zeppelin.pyspark.useIPython	Set to true to use the IPython, else set to false.

7.3.13 spark Interpreter

The configurations for the spark interpreter are given as follows:

Table 13 spark Interpreter

Field	Description
pgx.baseUrl	Enter the pgx.baseUrl URL in this field. This is the location where the data is pushed. For example: <code>http://<HOSTNAME>:7007</code>
spark.executor.memory	Enter the amount of memory to use for the executor process.
spark.master	Enter the cluster manager to connect to. For example: <code>local[*]</code>
spark.yarn.archive	Enter the archive containing the required Spark jars for distribution to the YARN cache, to make Spark runtime jars accessible from YARN side.
spark.app.name	Enter the name of the application. For example: zeppelin
zeppelin.spark.ui.hidden	
zeppelin.spark.maxResult	Enter the maximum number of results that need to be fetched.
spark.pyspark.python	Enter the Python binary executable to use for PySpark in both driver and executors. For example: python
zeppelin.spark.enableSupportedVersionCheck	Set to true or false.
args	Enter the Spark commandline args.
zeppelin.spark.useNew	Set to true to use the new version of the SparkInterpreter.
zeppelin.spark.useHiveContext	Set to true to use HiveContext instead of SQLContext.

Table 13 spark Interpreter

Field	Description
zeppelin.spark.uiWebUrl	This value overrides Spark UI default URL. Note: The value must be a complete URL.
zeppelin.spark.printREPLOutput	Indicates to print the REPL output.
spark.cores.max	Enter the total number of cores to use.

8 Managing Tasks

This section includes the following topics:

- [Accessing Tasks](#)
- [Task Statuses](#)
- [Table Columns](#)
- [Table Filters](#)

8.1 Accessing Tasks

To access Tasks, follow these steps:

1. Navigate to the FCC Studio workspace.
2. Click the menu icon in the upper-left corner.

The menu items are listed.

3. Click **Tasks**.

The **Tasks** page lists all tasks of FCC Studio and displays the notebook, paragraph, interpreter, and user associated with each task.

Notebook	Paragraph	Interpreter	Status	Creation Time	Queue Time	Run Time	User
RMF Scenario - CU Focus	Alert Generation Condition on High Risk Customer	fcc-spark-scala	success	10 Feb 2020 03:49 pm	-	-	
RMF Scenario - CU Focus	Aggregate Transaction on Customer	fcc-spark-scala	success	10 Feb 2020 03:49 pm	-	-	
RMF Scenario - CU Focus	Event Creation	fcc-spark-scala	success	10 Feb 2020 03:49 pm	-	-	
RMF Scenario - CU Focus	Alert Generation Condition on Low Risk Customer	fcc-spark-scala	success	10 Feb 2020 03:49 pm	-	-	
RMF Scenario - CU Focus	Alert Result	fcc-spark-scala	success	10 Feb 2020 03:49 pm	-	-	
RMF Scenario - CU Focus	List Dataset	fcc-spark-scala	success	10 Feb 2020 03:49 pm	-	-	
RMF Scenario - CU Focus	Wire Transaction - Both Credit and Debit	fcc-spark-scala	success	10 Feb 2020 03:49 pm	-	-	
RMF Scenario - CU Focus	Monetary Instrument - Both Credit and Debit	fcc-spark-scala	success	10 Feb 2020 03:49 pm	-	-	

8.2 Task Statuses

The tasks can have the following possible statuses:

Table 1 Task Statuses

Field	Description
created	Indicates that the task is just created.
queued	Indicates that the task is in a queue, waiting to be run. This can happen when the same user runs multiple paragraphs of the same interpreter in the same notebook - the interpreter will first finish executing the first paragraph (i.e., task) and then moves onto the second one which will have status queued until then.
running	Indicates that the tasks are being executed.

Table 1 Task Statuses

Field	Description
rejected	Indicates that the task is rejected.
success	Indicates that the task is successfully completed.
cancelled	Indicate that the execution of the task is cancelled.(For example, by clicking the 'Cancel Execution' ("Stop") button on a paragraph).
error	Indicates that an error has occurred during execution of a task. The error can be one of the following: <ul style="list-style-type: none"> • The concerned interpreter is unsupported • The interpreterClient is disconnected • The task is not found • The status of the task cannot be changed to running or success

8.3 Table Columns

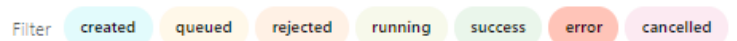
Table 2 Table Columns

Field	Description
Notebook	Indicates the name of the notebook for which the task was created for.
Paragraph	Indicates the title of the paragraph associated with the task, or if there is no title, the first line of code of the paragraph
Interpreter	Indicates the name of the interpreter that the task was created to run against
Status	Indicates the status of the task
Creation Time	Indicates the time (in device-local time) when the task was created
Queue Time	Indicates the total time spent by the task in the queue, that is, the time between the creation of the task and the beginning of the task execution
Run Time	Indicates the time taken for the task to run, that is, the time between the beginning and the end of the task execution
User	Indicates the username of the user who created the task

8.4 Table Filters

The list of tasks can be filtered in a number of ways to make it easy to search specific tasks. You can filter a task by the following categories:

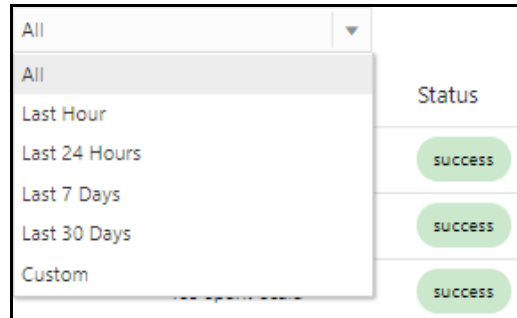
- **Task status:** Filters the tasks based on the statuses that match the selected statuses.



- **Text:** The search box allows the user to filter the tasks based on notebook name, paragraph title, paragraph code, interpreter, and user.



- **Date and Time:** Filters the tasks based on task creation time. The following options are available:



- Last Hour
- Last 24 Hours
- Last 7 days
- Last 30 days
- Custom: Allows the user to enter a custom from and/or to date-time. If empty, it assumes infinite past/future.

9 Managing Permissions

This section includes the following topics:

- [Permissions Overview](#)
- [Accessing Permissions Page](#)

9.1 Permissions Overview

This section provides information on users, groups, roles, and other permissions. It includes the following topics:

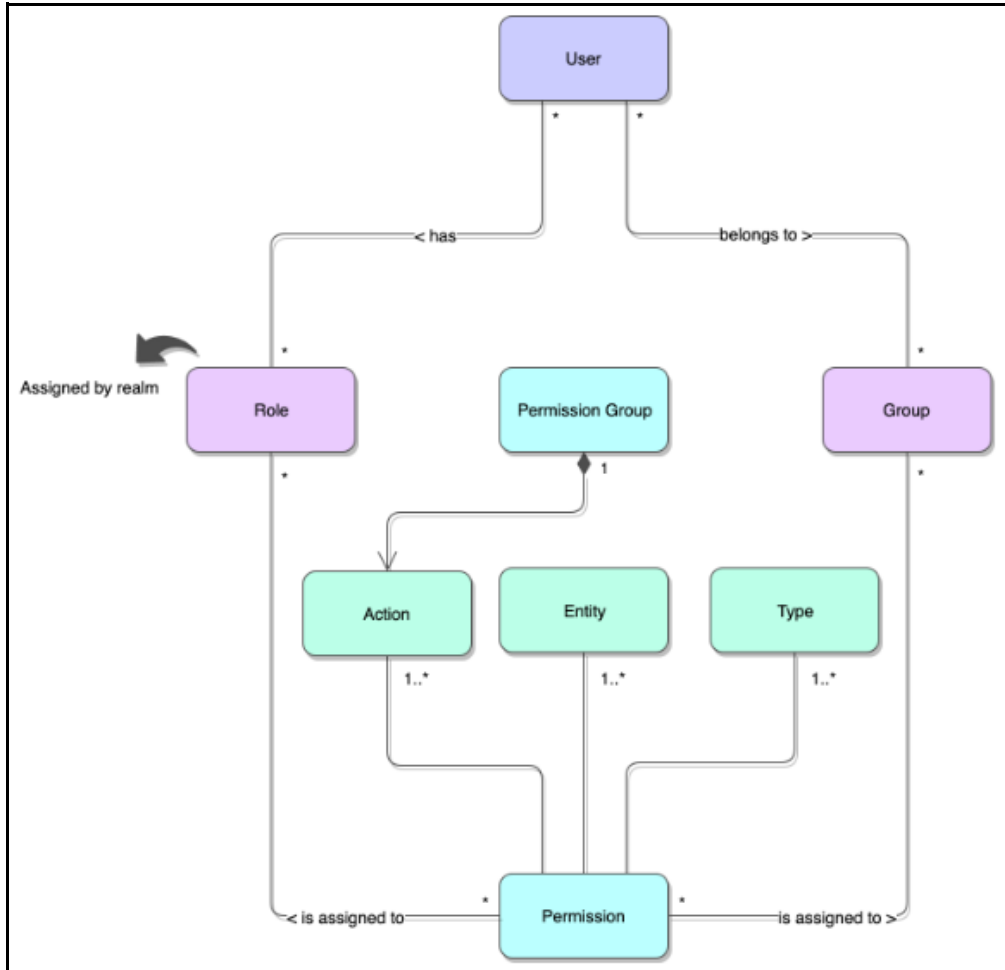
FCC Studio offers a rich permission system. The four key components are permissions, permission templates, roles, and groups.

A permission is an action that applies to entities and types. A permission can be a general action, an action on all entities of a particular type, or an action on a specific entity of a particular type.

For example:

- `general:create_notebook` - The general action to create a notebook.
- `graph:graph_update` - The action to update all entities of type graph, that is, the action to update all the graphs in the **Graphs** page.
- `notebook:export` - The action to export a specific entity of type notebook, that is, the action to export a particular notebook.

To evaluate whether a user can perform an action on an entity of a type, the permission must be assigned to the user, to one of their roles, or groups.



When sharing a notebook, you can select the user(s)/member(s) to share a notebook and grant the required permission.

The screenshot shows a "Share Notebook" dialog box. It has a title bar "Share Notebook" and a "Current Permissions" section. This section lists three entries:

- admin**: template, widget delete, view result + 26 more. Includes edit and delete icons.
- user**: layout, template, view code + 7 more. Includes edit and delete icons.
- user@oracle.com**: template, widget delete, view result + 26 more. Includes edit and delete icons.

Below this is an "Add new Permissions" section with a search input field and an "Add" button. At the bottom are "Cancel" and "Save" buttons.

Select Permissions

Permission-Group

read write own

limited_read

▲ Advanced Mode

<input type="checkbox"/> *	<input checked="" type="checkbox"/> attach	<input checked="" type="checkbox"/> clear
<input checked="" type="checkbox"/> clone	<input checked="" type="checkbox"/> detach	<input checked="" type="checkbox"/> export
<input checked="" type="checkbox"/> iframe	<input checked="" type="checkbox"/> layout	<input checked="" type="checkbox"/> restart_cluster
<input checked="" type="checkbox"/> run_all	<input checked="" type="checkbox"/> snapshot	<input checked="" type="checkbox"/> template
<input checked="" type="checkbox"/> toggle_show_code	<input checked="" type="checkbox"/> update	<input checked="" type="checkbox"/> view
<input checked="" type="checkbox"/> view_code	<input checked="" type="checkbox"/> view_result	<input checked="" type="checkbox"/> widget_create
<input checked="" type="checkbox"/> widget_delete	<input checked="" type="checkbox"/> widget_execute	<input checked="" type="checkbox"/> widget_modify
<input checked="" type="checkbox"/> widget_move	<input checked="" type="checkbox"/> widget_view	

9.2 Accessing Permissions Page

To access permissions, follow these steps:

1. Navigate to the FCC **Studio** workspace.
2. Click the menu icon in the upper-left corner.
The menu items are listed.
3. Click **Permissions**.

The **Permissions** page lists the following:

- [Users](#)
- [Groups](#)
- [Roles](#)
- [Permission Templates](#)

The screenshot shows the 'Permissions' page with three main sections: Users, Groups, and Roles. Each section has a table listing items and their associated permissions.

Username	Last Login	Roles	Groups	Permissions	Creation Time	Actions
admin@oracle.com	05 Sep 2019 05:21 pm	admin	-		05 Sep 2019 05:08 pm	
guest@oracle.com	-	-	-		05 Sep 2019 02:23 pm	
user10@oracle.com	-	-	-	view_sessions, view, paragraph_comment, export, iframe, paragraph_view, view_code, view_result, graph_view, export_all	05 Sep 2019 02:23 pm	

Name	Permissions	Creation Time	Actions
group1		05 Sep 2019 02:23 pm	
group2	view_result, clone, iframe, view, paragraph_comment, view_code, snapshot, paragraph_view, view_sessions, export, Show More ...	05 Sep 2019 02:23 pm	
group3	view_sessions, view, paragraph_comment, view_result, paragraph_view, export, view_code, iframe, graph_view, export_all	05 Sep 2019 02:23 pm	

9.2.1 Users

The Users section lists all the users, the date of their last login, and their roles, groups, other permissions. Users cannot be added or deleted in this section, but the groups they belong to can be updated.

This is a close-up of the Users table from the screenshot above, showing the same data points: Username, Last Login, Roles, Groups, Permissions, Creation Time, and Actions.

9.2.2 Groups

A Group consists of one or more permissions. A user can belong to multiple groups. For example, Oracle Labs, General, and External. The **Groups** section allows users to view and manage all groups. Groups can be added, updated, and deleted.

This is a close-up of the Groups table from the screenshot above, showing the same data points: Name, Permissions, Creation Time, and Actions.

9.2.3 Roles

A Role consists of one or more permissions. A user can have multiple roles that are typically assigned by the realm. For example, admin, user, and guest. All the listed roles can be added, updated, and deleted.

Roles			
Roles are sets of users. Roles can have permissions assigned to them just like permissions can be assigned to users. Users inherit the permissions from all the roles they have. Role assignment is controlled by the realm.			
Name	Permissions	Creation Time	Actions
admin	+	05 Sep 2019 02:23 pm	
user	view_result view_sessions paragraph_view snapshot iframe clone view export view_code paragraph_comment Show More ...	05 Sep 2019 02:23 pm	

9.2.4 Permission Templates

A Permission Template is a set of actions. For example, limited_read, read, and write. All the listed Permission Templates can be added, updated, and deleted.

Permission Templates			
Set of (often logically related) actions to simplify applying multiple permissions (e.g. when assigning permissions in the "Share Notebook" dialog)			
Name	Permissions	Creation Time	Actions
read	iframe view export paragraph_view view_code clone paragraph_comment view_sessions snapshot view_result Show More ...	05 Sep 2019 02:23 pm	
write	view_sessions toggle_show_code update paragraph_execute paragraph_create template clone paragraph_delete attach clear Show More ...	05 Sep 2019 02:23 pm	
own	+	05 Sep 2019 02:23 pm	
limited_read	iframe paragraph_comment view_code view_result view_sessions view export paragraph_view graph_view export_all	05 Sep 2019 02:23 pm	

10 Managing Credentials

FCC Studio provides a secure and safe credential management. For example, passwords, Oracle Wallets, or KeyStores. The credentials can be used to connect to data sources.

This section includes the following topics:

- [Accessing Credentials](#)
- [Using Credentials](#)

10.1 Accessing Credentials

To access credentials page, follow these steps:

1. Navigate to the FCC Studio workspace.
2. Click the menu icon in the upper-left corner.
The menu items are listed.
3. Click **Credentials**.

Name	Type	Content	Creation Date
My Password	Password	*****	02 Dec 2019
My Wallet	Oracle Wallet	wallet_DB201902061356.zip	03 Dec 2019
Keystore	KeyStore file	keystore.jks	03 Dec 2019
Properties	Credential file	test.zip	03 Dec 2019
DB Password	Password	*****	03 Dec 2019

4. Click **Create** on the top right corner to create a new credential. The maximum file size allowed for the credential file is 128Kb.

5. To download the credential files, click the credential file name.
6. To delete a credential, click the trash bin button in the last column of the row of the credential.

10.2 Using Credentials

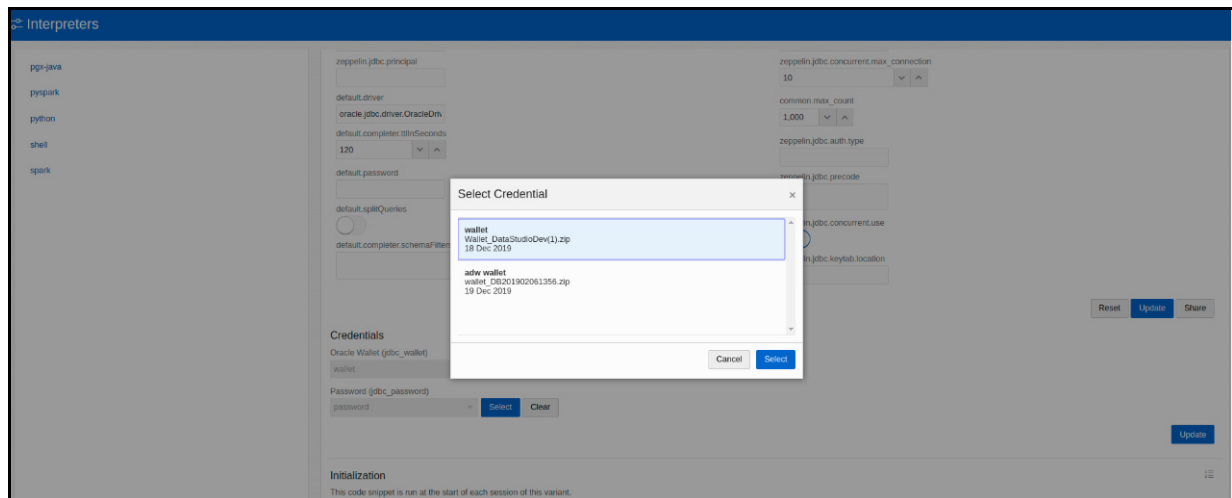
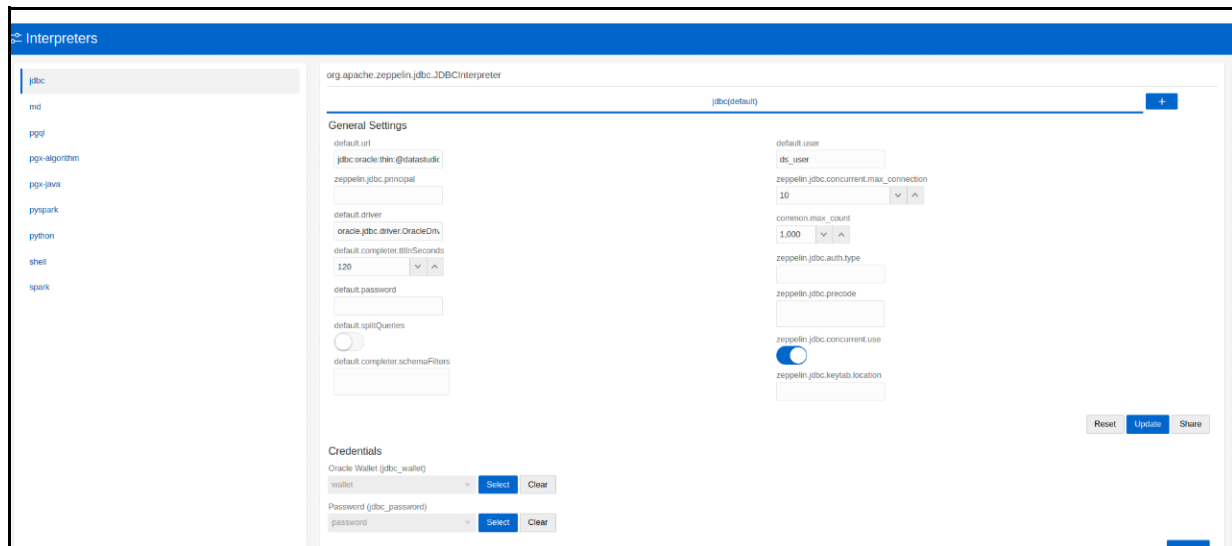
10.2.1 Link Credentials to Interpreter Variants

You can use your credentials and link them to certain interpreter variants to enable secure data access. If an interpreter variant is enabled to accept credentials, the **Credentials** section is displayed in the **Interpreters** page.

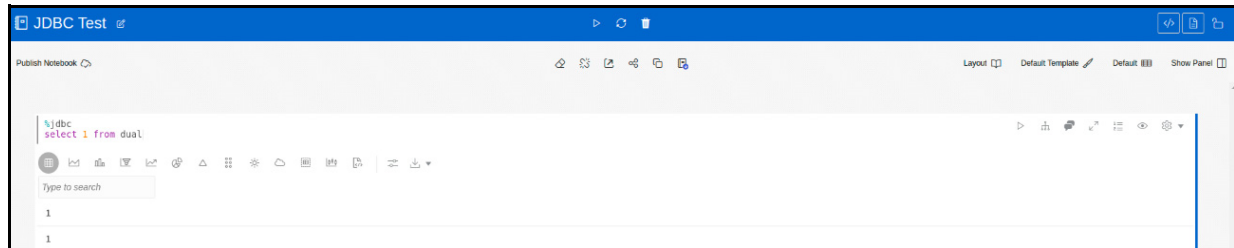
To link a credential to an interpreter variant, follow these steps:

1. Navigate to the **Interpreters** page
2. Select the credential that you want to link from the displayed popup and Click **Select**.
3. After you have linked your credentials, click **Update** to save the changes.

For example, you can link credentials (a wallet and a password) to JDBC interpreter variants. Link your wallet and password credentials to connect to an Oracle DB.



Use the JDBC interpreter to connect to the specified DB.



An Oracle Wallet provides a simple and easy method to manage database credentials. The wallet zip file includes the following:

- cwallet.sso
- ewallet.p12
- tnsnames.ora (optional)

11 Configuring ETL

- [Performing Data Source Configuration](#)
- [Performing Graph Configuration](#)

11.1 Performing Data Source Configuration

To configure a new data source for a graph, follow these steps:

1. Navigate to the `fcc_studio_etl_queries` table in the Studio Schema.
The FCDM related nodes and edges are available in the table.
2. If you want to add additional nodes or edges, you can add a new entry in the `fcc_studio_etl_queries` table.
3. Enter the following details in the `fcc_studio_etl_queries` table to add a new node or edge:

Table 1 `fcc_studio_etl_queries` Table Details

Column Name	Description	Applicable For
Type	Indicates the column name. Enter the value as NODE or EDGE.	Applicable for node and edge queries.
DF_NAME	Indicates the name for the node or edge.	Applicable for node and edge queries.
SOURCE	Indicates the source of the data. For example: FCDM or ICIJ	Applicable for node and edge queries.
DATAFRAME	Indicates the properties of the node or edge. NOTE: Enter this value only if the data source is Hive and not .a .csv file.	Applicable for node and edge queries.

Table 1 fcc_studio_etl_queries Table Details

Column Name	Description	Applicable For
QUERY	<ul style="list-style-type: none"> If the source is Hive, provide the Hive query. If the source is a .csv file, provide the query in the following format: <pre>spark.read.format("csv").option("header", "true").option("mode", "DROPMALFORMED").load("##FILE-PATH##").select("node_1", "node_2", "rel_type", "SourceID").withColumn("Label", lit("address of")).withColumnRenamed("node_1", "from").withColumnRenamed("node_2", "to").withColumnRenamed("rel_type", "EDGE_TYPE").withColumnRenamed("SourceID", "Source").filter(col("EDGE_TYPE") === "registered_address").withColumn("node_ID", concat(lit("#NUMBER #"), col("node_ID")))</pre> <p>For information on the query parameters, see Table 2, "Query Parameter Details".</p> <p>NOTE: Ensure that the source .csv file is UTF-8 compatible.</p>	Applicable for node and edge queries.
KEY_COLUMN_NAME	Set the value to the column name of your unique identifier, if the query is for node. For example: 'node_id'.	Applicable for node query.
SOURCE_NODE	Provide the DF_NAME of the node from which the edge starts from.	Applicable for edge query.
DESTINATION_NODE	Provide the DF_NAME of the node from which the edge ends to.	Applicable for edge query.
SOURCE_KEY_COLUMN_NAME	Set the value to the column name which has key_column values of the Source Node. For example: 'from_id'	Applicable for edge query.
DESTINATION_KEY_COLUMN_NAME	Set the value to the column name which has key_column values of the Destination Node. For example: 'to_id'	Applicable for edge query.
ACTIVE	Expected values: 'Y'/'N'. Set the value to Y to consider ETL and Graph loading.	Applicable for node and edge queries.

Table 2 Query Parameter Details

Query Parameter	Description
spark.read.format("csv")	Indicates the input file format. For example: .csv.

Table 2 Query Parameter Details

Query Parameter	Description
option("header", "true")	Indicates the presence of a header in the input file. <ul style="list-style-type: none"> • true indicates that the header is available in the input file. • false indicates that the header is absent in the input file.
load("Path").	<ul style="list-style-type: none"> • Load indicates to load the data from the mentioned file path. • Path indicates the path where the files are placed. You can load to multiple paths using the following format: ("Path1", "Path2", ...)
select("Col1","Col2","Col3","Col4")	Indicates the columns to be selected in the input file.
withColumn("A",lit("Test1"))	Indicates to add a new column with column name A and column value Test1.
withColumnRenamed("A","B")	Indicates to rename a column with a different name. For example: rename column from A to B.
filter(col("A")==="Test1")	Indicates the "Where" filter condition. Here, the value for column A is Test1.
withColumn("B",concat(lit("Test1"),col("A")))	Indicates to add a new column B, whose value is the concatenated value of Test1 and column A. For example: Test1=ABC Column A contains Country and Pincode as the column values. Column B gets ABCCountry and ABCPincode as column values.

4. If the source is a .csv file, configure the file path in the `fcc_studio_etl_files` table.

NOTE Ensure that the source .csv file is UTF-8 compatible.

5. Enter the following details in the `fcc_studio_etl_files` table to add file path:

Table 1 fcc_studio_etl_files Table Details

Column Name	Description
DF_NAME	Indicates the name of the node or edge.
DF_SEQ_NO	Indicates the unique sequence ID for each file.
FILEPATH	Enter the path where the .csv files are stored. NOTE: If one data frame has multiple .csv files, then make separate entries for all the files. For example: see Figure 1, "fcc_studio_etl_files Table" .
FILEORDER	If data must be imported from multiple files, specify the order in which the files must be read.

Figure 1: fcc_studio_etl_files Table

DF_NAME	FILEPATH	DF_SEQ_NO	FILE_ORDER
1 Offshore_edges_is_related_to	12	1
2 Bahama_External_Address	13	1

11.2 Performing Graph Configuration

- Attributes Case in Graph
- Extra Empty Nodes and Edges Providers
- Additional Configuration

11.2.1 Attributes Case in Graph

The rules applied on the attribute names are as follows:

1. Attribute name are split on the basis of '_'. The '_' acts as a separator and the text that is separated by the separator are split into words.
2. The first alphabet of each word is set to uppercase and remaining alphabets are set to lowercase.

For example:

- Attribute Name: sample_attribute
- Renamed Attribute Name: Sample Attribute

If any specific attribute name is required in the graph, you must update the same in the FCC_GRAPH_COLUMN_NAME_MAPPING table:

Where,

COLUMN_NAME: Indicates the attributes name in queries

RENAMED_COLUMN_NAME: Indicates the required attribute name

COLUMN_DATA_TYPE: Indicates the PGX's datatype of the attribute

NOTE

- The accepted PGX's datatype formats are boolean, integer, float, long, double, string, date, local_date, time, timestamp, time_with_timezone, timestamp_with_timezone, and point2d.
- The date is deprecated, hence you can use one of following instead:
 - local_date
 - time
 - timestamp
 - time_with_timezone
 - timestamp_with_timezone

For example, if the values are as follows:

- COLUMN_NAME: sample_attribute

- RENAMED_COLUMN_NAME: Sample_AttributeName
- COLUMN_DATA_TYPE: string

Then the attribute name shown in graph is, Sample_AttributeName

Figure 2: FCC_GRAPH_COLUMN_NAME_MAPPING Table

	◇ COLUMN_NAME	◇ RENAMED_COLUMN_NAME	◇ COLUMN_DATA_TYPE
1	original_id	Original ID	string
2	tax_id	Tax ID	string
3	debit_or_credit	Debit or Credit	string
4	initialShowPropName	initialShowPropName	string

11.2.2 Extra Empty Nodes and Edges Providers

Currently, heterogeneous graph doesn't support dynamic addition of Nodes and Edges Provider in the graph. If extra nodes or edge providers are required, then you must add the entries to the FCC_GRAPH_EMPTY_ENTITY_MAPPING table.

Where,

- TYPE: Indicates the type of empty entity provider to be added.
Expected value: "NODE" or "EDGE"
- NAME: Indicates the name of the entity provider.
- COLUMN_MAPPING: Indicates the attributes required for the entity with its datatype. The value must be comma separated paired value of column name and its type.

For example: column1:string,column2:long

NOTE

- In case of NODE, do not specify key_column for the node. In case of EDGE, do not specify source and destination key_columns.
- The accepted PGX's datatype formats are boolean, integer, float, long, double, string, date, local_date, time, timestamp, time_with_timezone, timestamp_with_timezone, and point2d.
- The date is deprecated, hence you can use one of following instead:
 - local_date
 - time
 - timestamp
 - time_with_timezone
 - timestamp_with_timezone

- Example 1:
 - TYPE: NODE
 - NAME: extra_node
 - COLUMN_MAPPING: name:string,phone_number:integer

Here, an extra vertex provider with the name "extra_node" is added with the attributes, Name and Phone Number, datatype, string, and integer respectively.

- Example 2:
 - TYPE: EDGE
 - NAME: extra_edge
 - COLUMN_MAPPING: name:string,risk:long,edge_type:string

Here, extra edges will be formed between every node providers including itself with the name as "<source_node_provider>_extra_edge_<destination_node_provider>", with the attributes, Name, Risk and Edge Type, datatype, string, long, and string respectively.

Figure 3: FCC_GRAPH_EMPTY_ENTITY_MAPPING Table

NAME	TYPE	COLUMN_MAPPING
1 searched_entity	NODE	source:string,label:string,name:string,address:string,tax_id:string,date:string,initialShowPropName:string
2 is similar to	EDGE	label:string,match weight:float,match score:string

11.2.3 Additional Configuration

For the Out- of-the-box (OOB) graph's configuration, the following parameters are set in the FCC_DATASTUDIO_CONFIG table:

1. local_date_format:

Default value: ["M/d/yyyy", "yyyy-MM-dd"].

NOTE

- Include '[', ']', '' and ','.
- The date format option can be used only to view the date type of an attribute on the graph, in the configured format.

2. vertex_id_type:

Default value is "long" as per the OOB queries.

This parameter represents the datatype of vertex_id column or key_column of node providers.

NOTE

This datatype should be consistent across all nodes.

Figure 4: FCC_DATASTUDIO_CONFIG Table

PARAMNAME	PARAMVALUE	DESCRIPTION
1 VERTEX_ID_TYPE	long	(null)
2 LOCAL_DATE_FORMAT	["M/d/yyyy", "yyyy-MM-dd"],	(null)

12

Appendix A - Creating and Executing Run Executable

NOTE

Ensure that the username and password are set before executing a notebook batch. For more information, see [Preparing for Batches](#).

To create and execute run executable, follow these steps:

1. Log in to the OFSAA application with a user who has the privilege to create run executable.
2. Select **Financial Services Anti Money Laundering** from the tiles menu.

The Financial Services Anti Money Laundering Application Home Page is displayed with the Navigation list to the left.

3. Navigate to **Common Tasks > Rule Run Framework > Run** from the navigation list.

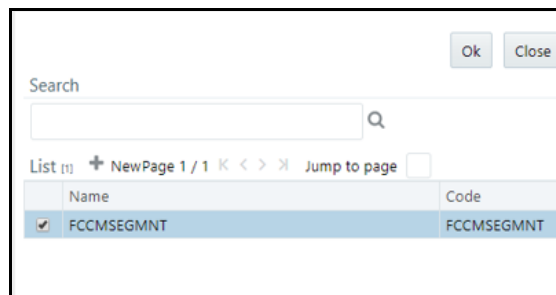
The **Run Definition** page is displayed.

4. Click **New** on the List toolbar.

The **Rule Run Framework** window is displayed.

5. Under the **Linked To** toolbar, click the button next to **Folder**.

The **Folder Selector** dialog box is displayed.

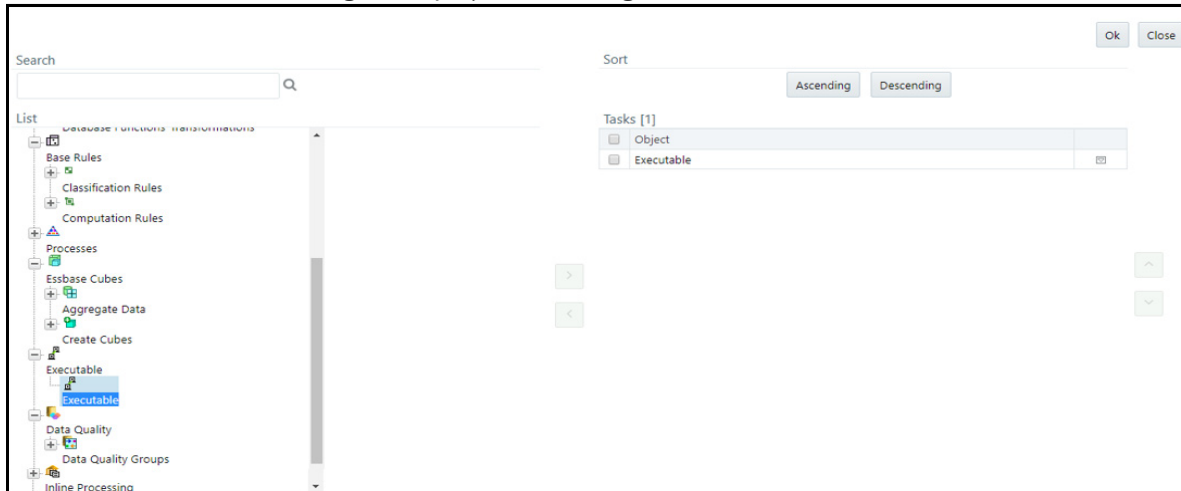


6. Select the folder that is to be linked to the run executable.
7. Click **OK**.
8. Enter the following details in the **Master Information** toolbar.

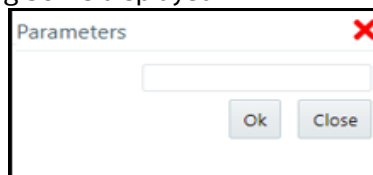
Table 1 Adding Run Definition

Field	Description
Code	Enter the Code of the process.
Name	Enter the Name of the process.
Type	Select Type for the process.

9. Click **OK**.
10. Click **Selector** on the List toolbar. From the options displayed, select **Job**.
The **Jobs** page is displayed.
11. Click **Executable** on the list. From the options displayed, select **Executable**.
The **Executable** gets displayed on the right.



12. Select **Executable** from the Tasks list, click the button next to the Executable option.
The **Parameters** dialog box is displayed.



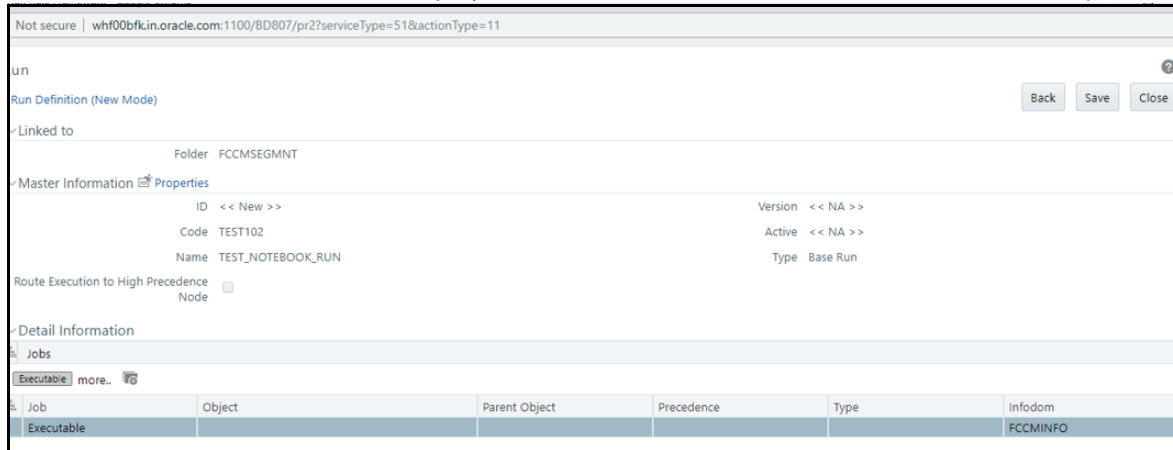
13. Enter the parameters in the following format to create run executable:

Table 1 List of Parameters for Batch Execution

SL. No.	Command Format		Description
1	"FCCM_Studio_SqoopJob.sh", "batchId", "ficmisdate", "userparams", "source for datamovement", "source to include in graph"		Use this command for data movement and graph loading.
	File Name	Parameters	Description
	FCCM_Studio_Sqoop-Job.sh		Used for data movement and graph loading in Big Data Environment
		batchId	Indicates the ID of the batch that you want to execute.
		ficmisdate	Indicates the date on which the source data is available.
		userparams	Includes the data movement code and snapshot date.
		source for datamovement	Indicates the source data type for data movement.
		source to include in graph	Indicates to source to be included in graph. The values must be comma separated.
2	"FCCM_Studio_NotebookExecution.sh", "notebookID", "outputParagraphID", "scenarioID", "thresholdsetID", "extraparams"		Use this command for executing published notebook.
	File Name	Parameters	Description
	FCCM_Studio_NotebookExecution.sh		Used for batch execution of published notebook.
		notebookId	Indicates the ID of the required notebook
		outputParagraphId	Indicates that the value is always "null"
		scenarioId	Indicates the ID of Scenario
		thresholdsetId	Indicates the ID of the threshold set with which notebook will run
		sessionId	Indicates the ID of the session in which notebook will run
		extraparams	For scenario notebook, it will be "null", but for notebook execution, it depends on the paramkeys used in the notebook

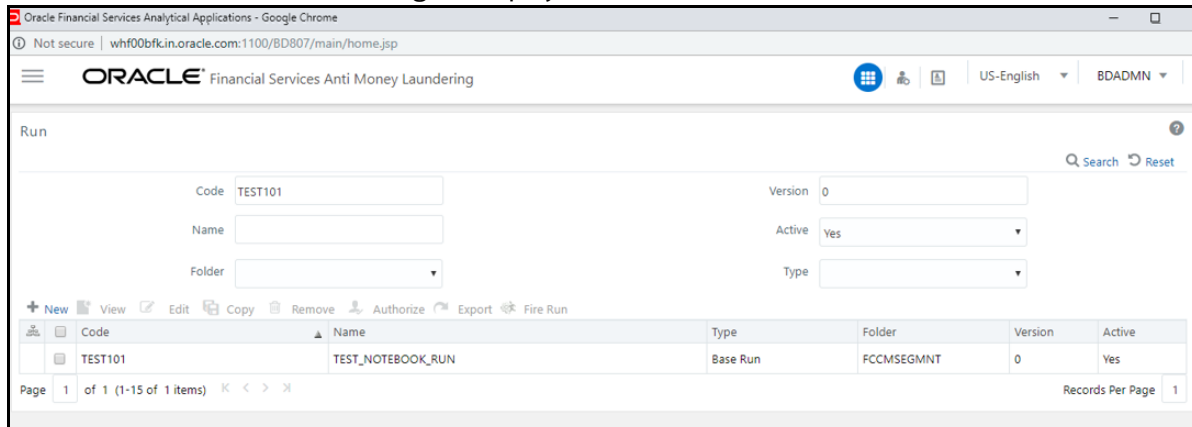
14. Click **OK**.

The run executable is displayed in the Detail Information section on the **Run Definition** page.



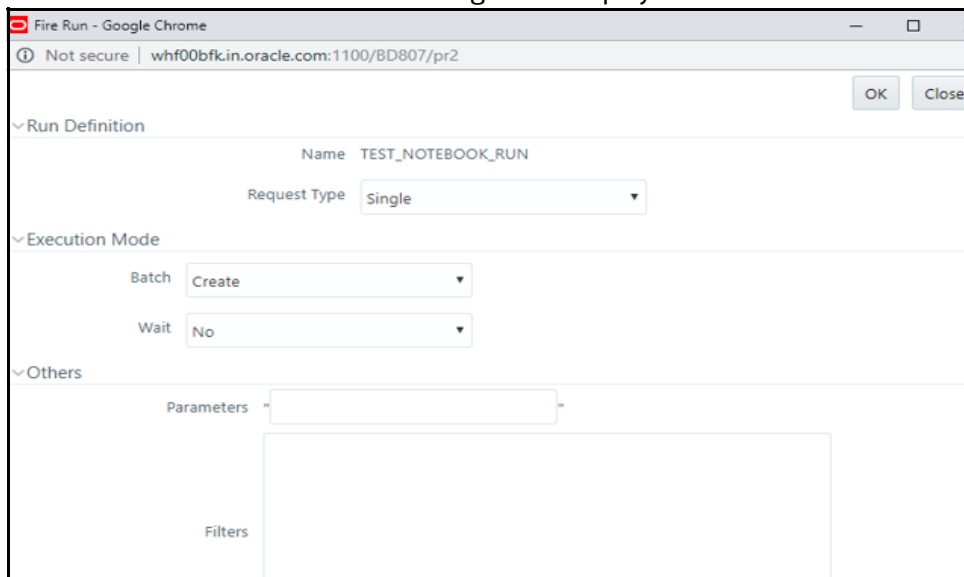
15. Click **Save**.

A confirmation message is displayed. The Run executable is created.



16. Select the newly created run executable from the **Run Definition** page that is to be created and click **Fire Run**.

The **Fire Run Rule Framework** dialog box is displayed.



17. Enter the following details:

Table 1 Adding Fire Run Details

Field	Description
Request Type	Select Request Type based on the following options: <ul style="list-style-type: none"> • Single: If the batch has to be executed once. • Multiple: If the batch has to be executed multiple times at different intervals.
Batch	Select Batch. It has the following options: <ul style="list-style-type: none"> • Create • Create and Execute From these options, select Create & Execute.
Wait	Select Wait. It has the following options: <ul style="list-style-type: none"> • Yes: This will execute the batch after a certain duration. Enter the duration as required. • No: This will execute the batch immediately.
Filters	Enter the filter details.

18. Click **OK** to run the Run Executable.

The Run executable starts executing.

19. From the Navigation List, navigate to **Common Tasks**, click **Operations**, and then select **Batch Monitor**.

The **Batch Monitor** window is displayed.

20. Select the batch that is run in step 18. Select the Information Date and Batch Run ID from the drop-down.

21. Click on **Start Monitoring** in Batch Run Details.

The Batch Run ID and Batch Status details are displayed in the Batch Status details section.

13 Appendix B - Adding Packages to Python Interpreter

To add packages to the python interpreter in FCC Studio, follow these steps:

1. Create a folder at the same location as the `<Studio_Installation_Path>`, and create a file inside the folder with the file name, `Dockerfile`.
2. Copy and paste the following content as a template into the `Dockerfile` file:

```
FROM ofsaa-fccm-docker-release-local.dockerhub-den.oraclecorp.com/fcc-studio/fcc-python:8.0.7.3.0
```

```
USER root
```

```
RUN pip3 install scipy pandas cx_oracle
```

```
USER interpreteruser
```

3. Modify the `Dockerfile` file depending on the following installation method:

- a. If Internet connectivity is available, follow these steps:

Depending on the version of the Python package, install the Python packages, `scipy` and `cx_oracle` as follows:

```
RUN pip install scipy cx_oracle
```

or

```
RUN pip3 install scipy cx_oracle
```

NOTE Ensure to use `pip` for Python2, and `pip3` for Python3..

- b. If Internet connectivity is unavailable, and you have downloaded files of Python packages, follow these steps:

- i. Place the downloaded file(s) in the folder beside the `Dockerfile` file.

- ii. Execute the shell command to install the offline Python packages.

For example, to install using Python3:

```
RUN pip3 install --no-deps numpy-1.17.4-cp36-cp36m-manylinux1_x-86_64.whl
```

- c. Build the docker image using the following command:

```
docker build . --build-arg http_proxy=http://<proxy-url>:<port> --build-arg https_proxy=http://<proxy_url>:<port> -t <my.docker-registry.com:port>/ofsaa-fccm-docker-release-local.dockerhub-den.oraclecorp.com/fcc-studio/fcc-python:<version>
```

Where:

`<my.docker-registry.com:port>` is the docker-registry url with port number and `<version>` is the custom tag for this image.

For example:

```
docker build . --build-arg http_proxy=http://my-proxy-url:80 --build-arg https_proxy=http://my-proxy-url:80 -t my.docker-registry.com:5000/ofsaa-fccm-docker-release-local.dockerhub-den.oraclecorp.com/fcc-studio/fcc-python:8.0.7.3.0-C1
```

-
- d. Push the images using the following command:
- ```
docker push <my.docker-registry.com:port>/ofsaa-fccm-docker-release-
local.dockerhub-den.oraclecorp.com/fcc-studio/fcc-python:<version>
```
4. Update the image name in the `fcc-python.yml` file available in the `<Studio_Installation_Path>/deployments/fcc-python.yml` directory:
- ```
spec:  
  containers:  
    - name: python-interpreter  
      image: ofsaa-fccm-docker-release-local.dockerhub-den.oracle-  
corp.com/fcc-studio/fcc-python:<version>
```
5. Restart the FCC Studio application.
- Execute the following command from the Kubernetes master node:

```
kubectl delete namespace fccs
```
 - Navigate to the `<Studio_Installation_Path>/bin` directory.
 - Execute the following command:

```
./fcc-studio.sh --registry <registry URL>:<registry port>
```

The desired packages are added to the python interpreter.

