

**Oracle® Financial Services Crime and  
Compliance Studio**

Administration and Configuration Guide

Release 8.0.7.1.0

**E91246-01**

October 2019

Copyright © 2019 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, 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 on 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. 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.

# Contents

<b>Document Control</b> .....	vii
<b>About this Guide</b> .....	ix
Summary .....	ix
Audience.....	ix
Related Documents .....	ix
<b>1 About Oracle Financial Services Crime and Compliance Studio</b>	
<b>Introduction to Crime and Compliance Studio</b> .....	1-1
<b>Salient Features of Crime and Compliance Studio</b> .....	1-1
<b>Architecture of Crime and Compliance Studio</b> .....	1-2
.....	1-2
<b>Oracle Financial Crime Graph Model</b> .....	1-2
<b>2 Managing User Administration</b>	
<b>Managing Identity and Authorization</b> .....	2-1
Identity and Authorization Process Flow.....	2-1
Creating and Authorizing User .....	2-2
Mapping User with User Group.....	2-2
<b>Granting Permissions</b> .....	2-2
<b>3 Accessing Crime and Compliance Studio</b>	
<b>Accessing Crime and Compliance Studio Application</b> .....	3-3
<b>4 Managing Studio Batches</b>	
<b>Preparing for Batches</b> .....	4-1
<b>Performing Batches</b> .....	4-1
Data Movement and Graph Loading for Big Data Environment.....	4-1
Executing Published Scenario Notebook.....	4-2
<b>5 Configuring Security for PGX</b>	
<b>Prepare Certificates</b> .....	5-1
Create a Self-Signed Server Certificate .....	5-1
Create New Keystore .....	5-2
Extract the Certificate .....	5-2
Import Existing Certificate or Install Certificate from a Certificate Authority .....	5-3
<b>Prepare Client Keystore</b> .....	5-3
<b>Prepare Client Truststore</b> .....	5-4
<b>Configure PGX Server</b> .....	5-4
<b>Test Connection Using PGX Client Shell</b> .....	5-4

## 6 Configuring Interpreters

<b>Accessing Interpreters</b> .....	6-1
<b>Create New Interpreter Variant</b> .....	6-2
Create New JDBC Interpreter Variant .....	6-3
<b>Configure Interpreters</b> .....	6-3
fcc-jdbc Interpreter .....	6-4
fcc-ore Interpreter .....	6-5
fcc-pyspark Interpreter .....	6-6
fcc-python Interpreter .....	6-7
fcc-spark-scala Interpreter .....	6-8
fcc-spark-sql Interpreter .....	6-9
md Interpreter .....	6-10
pgx Interpreter .....	6-10

## 7 Configuring Data Sources for Graph

### A Creating and Executing Run Executable

### B Studio Services

<b>ETL Service</b> .....	B-11
graph.config.template.json File Details .....	B-11
etl.properties File Details .....	B-12
Fixed Set of Properties .....	B-12
Dynamically Named Properties .....	B-15
Sample for etl.properties File .....	B-16
<b>Data Forwarding Service</b> .....	B-22
forwarderServer.properties File Details .....	B-22
Graph Configuration File Tokens .....	B-23
Sample for forwarderServer.properties File .....	B-23
<b>Cross Language Name Matching Service</b> .....	B-24
NameMatchingLocations.properties File Details .....	B-24
Sample for NameMatchingLocations.properties File .....	B-25

---

---

# Document Control

<b>Version Number</b>	<b>Revision Date</b>	<b>Changes Done</b>
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.
8.0.7.1.0	Updated: October 2019	Updated the guide for 8.0.7.1.0 release.



---

---

# About this Guide

## Summary

This guide describes the physical and logical architecture of the Crime and Compliance Studio application. It also provides instructions for maintaining and configuring Studio, its subsystem components, and any third-party software required for operation.

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

## Audience

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

- **Implementation Consultant:** Installs and configures the Crime and Compliance Studio application at a specific deployment site, installs and upgrades any additional Oracle Financial Services solution sets, and requires access to deployment-specific configuration information. For example, machine names and port numbers.
- **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.

## Related Documents

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





---

---

# About Oracle Financial Services Crime and Compliance Studio

This chapter provides complete functional details about Oracle Financial Services Crime and Compliance Studio application.

This chapter includes the following sections:

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

## Introduction to Crime and Compliance Studio

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

Oracle Financial Services Crime and Compliance Studio is an integrated and comprehensive analytics toolkit designed to rapidly discover and model new financial crime patterns. Studio interacts with the database, processes the data, and generates patterns in various formats using interpreters. The Studio provides secure access to an institution's financial crime data with pre-defined scenarios, out-of-the-box graph queries, and visualizations.

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

The Studio uses Graph Analytics and Graph Query methods to analyze historic data available in the database, and forecast the generated patterns using various interpreters. Studio also uses Machine Learning Algorithms to gain insights from historical alert data, in order to prioritize the alerts generated by the detection engines.

Studio uses interpreters such as PGX, PGQL, GreenMarl, OFSAA Interpreter, OFSAA SQL Interpreter, and Markdown to generate patterns in various formats such as Table, Area Chart, Bar Chart, Funnel Chart, Line Chart, Pie Chart, Pyramid Chart, Tree Map, Sun Burst, Tag Cloud, Box Plot Chart, Scatter Plot Chart, and Raw Code.

## Salient Features of Crime and Compliance Studio

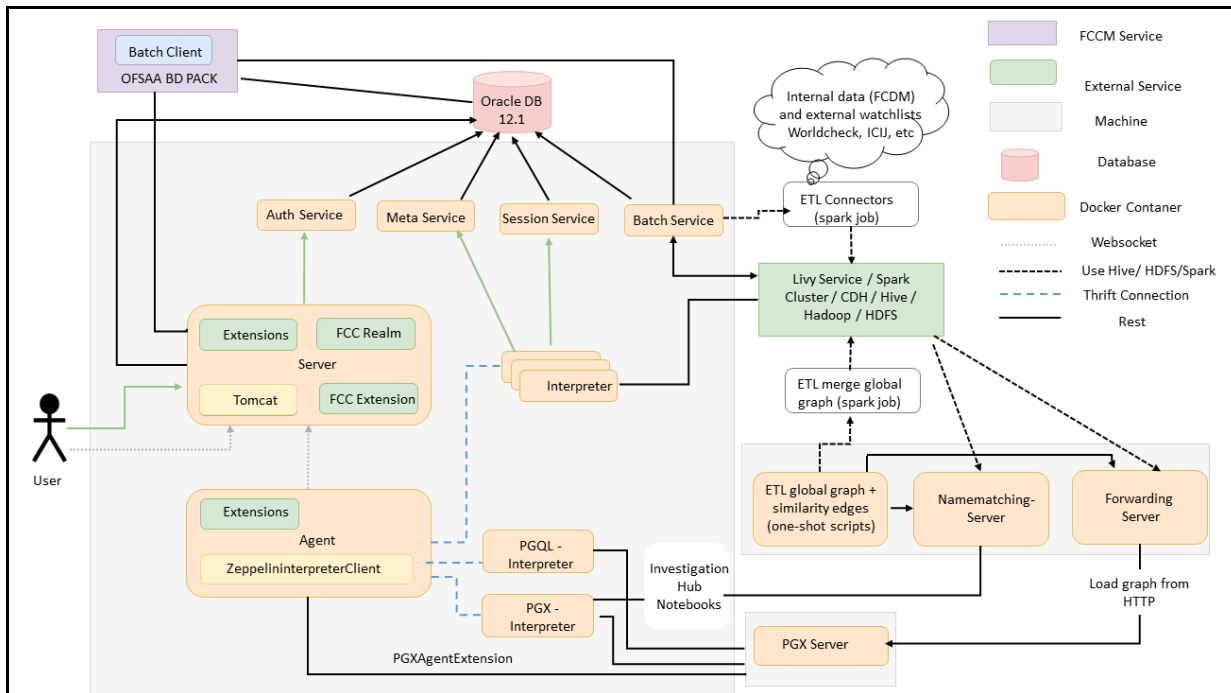
The salient features of Studio are as follows:

- Offers a unified tool for Graph Analytics, Data Visualization, Machine Learning, Scenario Authoring, Pattern Discovery, Data Mashups and testing for financial crime data
- Engineered to work with Apache Spark, the most prevalent analytics engine on Big Data
- Engineered to work 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 frameworks
- Engineered to work 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 mash-up Studio data with third-party data for discovery and modeling

## Architecture of Crime and Compliance Studio

The following diagram depicts the architecture of the Studio application:

**Figure 1–1 Crime and Compliance Studio Architecture**



## Oracle Financial Crime Graph Model

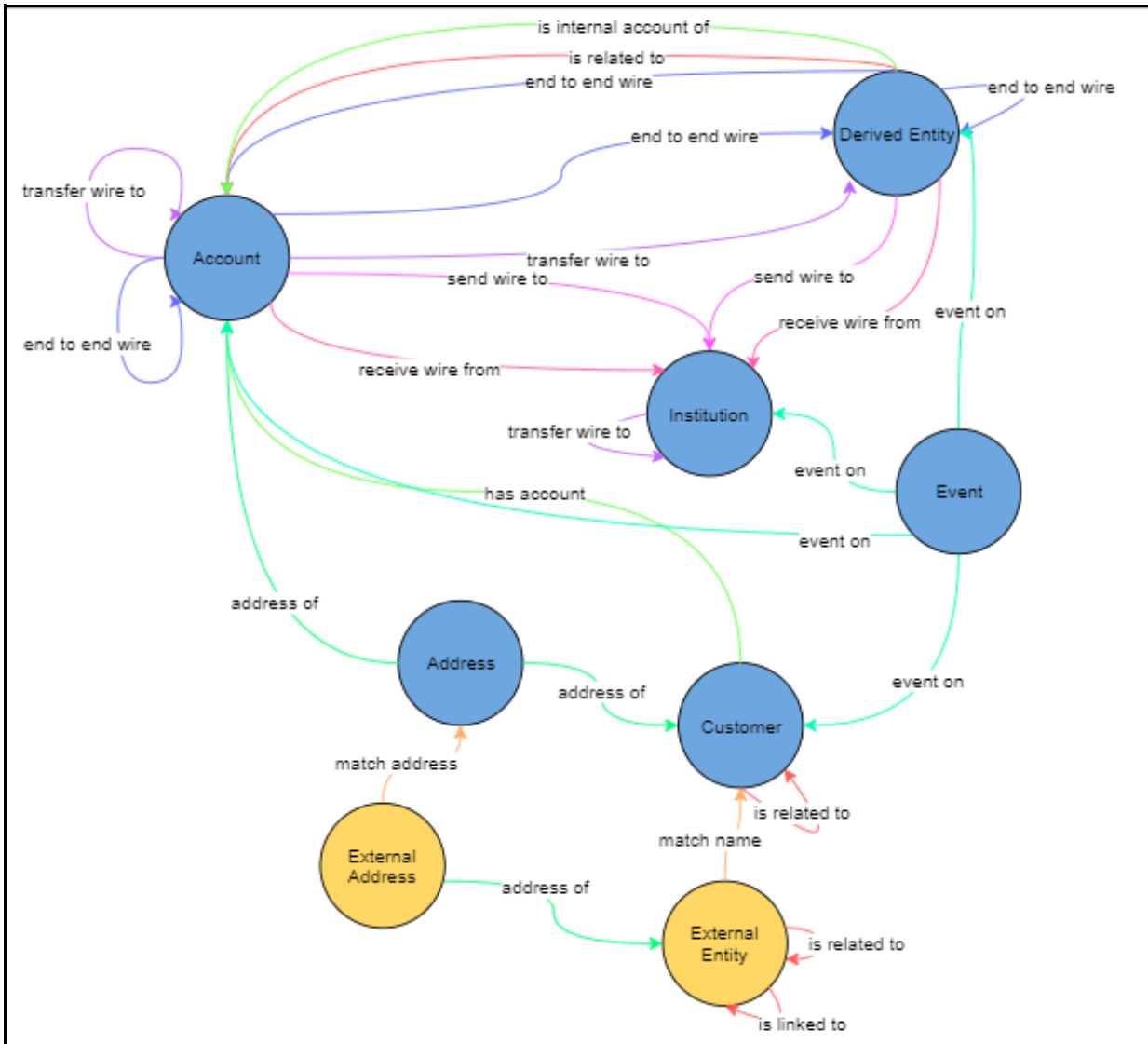
The Oracle Financial Crime Graph Model serves as a window into the financial crimes data lake. It collates disparate data sets into an enterprise-wide global graph, enabling a whole new

set of financial crime use cases. The Graph model enables to accelerate financial crime investigation use cases.

For information on Graph Data Model, see [Graph Data Model](#).

For information on the node and edge properties of the Oracle Financial Crime Graph Model, see [Data Model Guide](#).

**Figure 1-2 Oracle Financial Crime Graph Model**





---

---

## Managing User Administration

This chapter provides information on creating users who can access the Studio application and execute batches required for Studio. 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 Crime and Compliance Studio based on assigned roles.

The following topics are covered in this section:

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

### Managing Identity and Authorization

This section provides information on creating, mapping and authorizing users, and providing access to Studio application.

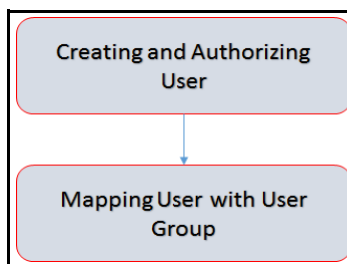
This section covers the following topics:

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

### Identity and Authorization Process Flow

[Figure 2-1](#) shows the process flow of identity management and authorization.

**Figure 2-1** *Managing Identity and Authorization Process Flow*



[Table 2-1](#) lists the various actions involved in the user administration process flow:

**Table 2–1 User Administration Process Flow**

Action	Description
<a href="#">Creating and Authorizing User</a>	Create a user by providing the user name, user designation, and the date during which the user is active in Studio.
<a href="#">Mapping User with User Group</a>	Map user with a user group that provides the user with the privileges of the mapped user group.

## Creating and Authorizing User

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

## Mapping User with User Group

This section provides information on mapping users with user groups. A user is mapped with a user group, and the user group is associated with a role. Each role comprises of certain predefined privileges.

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

[Table 2–2](#) describes the Roles and the corresponding User Groups in Studio.

**Table 2–2 Roles and User Groups in Studio**

Role	User Groups
DSADMIN	DSADMINGRP
DSINTER	DSINTERGRP
DSUSER	DSUSERGRP
DSBATCH	DSBATCHGRP

## Granting Permissions

1. Log in to Oracle Database from sys 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.

---

---

## Accessing Crime and Compliance Studio

### Accessing Crime and Compliance Studio Application

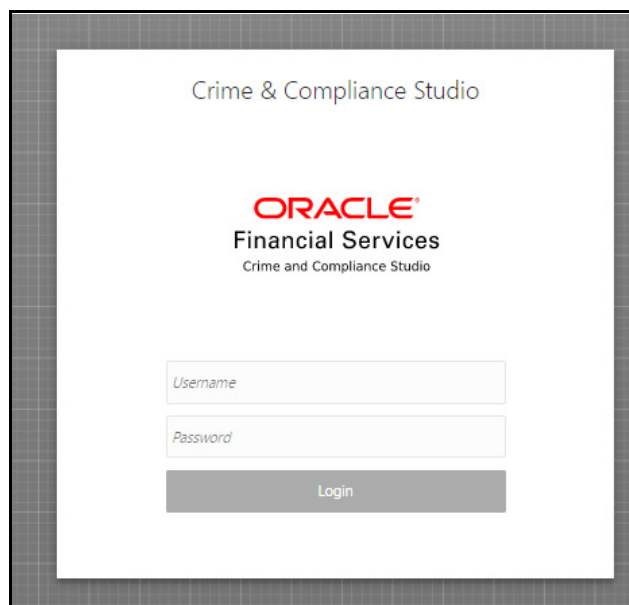
To access the Crime and Compliance Studio application as a system administrator:

1. Enter the Studio URL in your browser in the following format:

`https://<Host_Name>:7008`

The Crime and Compliance Studio Login page is displayed.

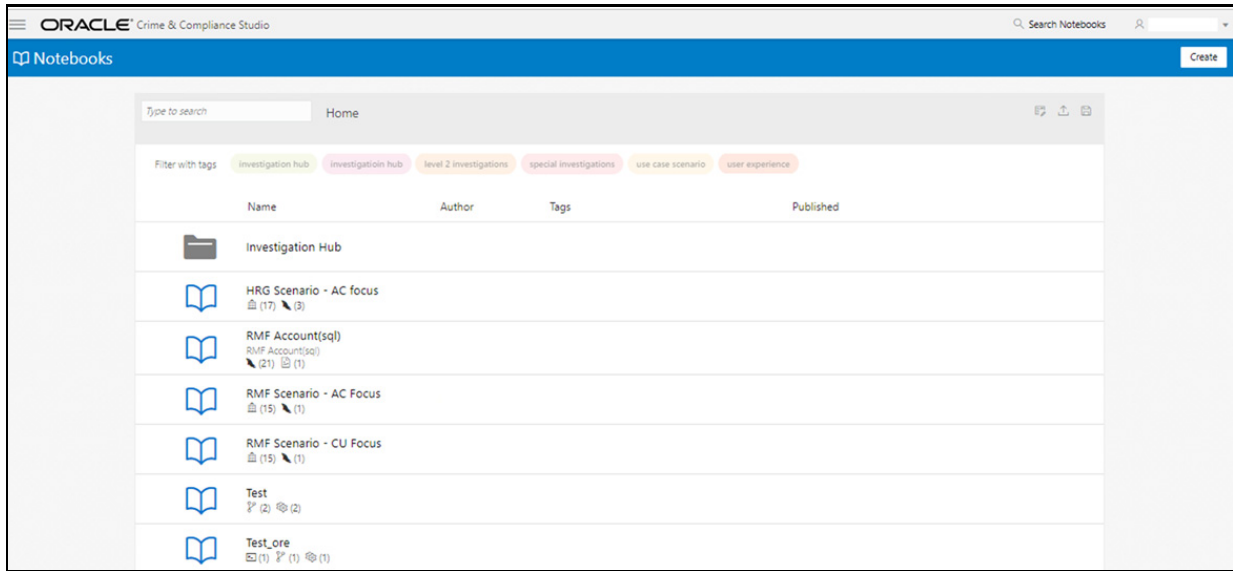
**Figure 3–1** Crime and Compliance Studio Login Page




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

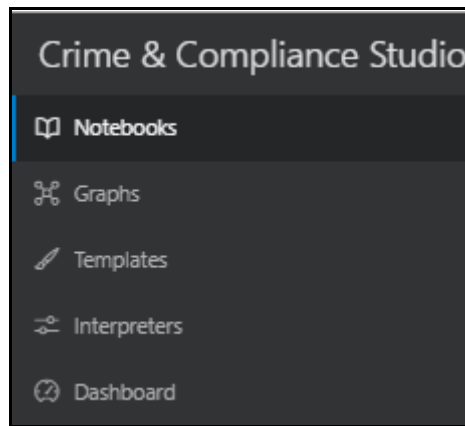
The Crime and Compliance Studio application's landing page is displayed.

**Figure 3–2 Crime and Compliance Studio Landing Page**



4. Click the  **Menu** icon on the top left corner.  
The menu items applicable to System Administrator are displayed.

**Figure 3–3 Menu Items**



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



---

---

## Managing Studio Batches

This chapter provides information on creating batches that are required for Crime and Compliance Studio. Batches are used for moving data, loading graphs, and running a notebook. This helps to move data from Oracle Database or Big data on a daily basis in Studio.

This chapter covers the following sections:

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

### Preparing for Batches

Before you perform the batches, follow these steps to prepare the batches:

1. Copy all the jars from `<STUDIO_INSTALLATION_PATH>/ficdb/lib` path into the `<FIC_HOME of OFSAA_Installed_Path>/ficdb/lib` path.
2. Copy the `NBExecutor.txt` file from the `<STUDIO_INSTALLATION_PATH>/ficdb/bin` path to the `<FIC_HOME of OFSAA_Installed_Path>/ficdb/bin` path.
3. Enter the following details in the `NBExecutor.txt` file:

```
username=<Studio Username>
```

```
password=<Studio Password>
```

---

---

**Note:** The username must be of a user who has permission to run Notebook.

---

---

### Performing Batches

The following are the types of batches that can be configured:

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

### Data Movement and Graph Loading for Big Data Environment

You can move data from *BD atomic* environment to *Big data* environment. As a result, a graph is created for the data in the form of the `.pgb` and `config.json` files. These files are used to view graphs and queries using PGQL and PGX.

For information on the Oracle Financial Crime Graph Model, see *The Manage Graphs* section in the *Oracle Financial Services Crime and Compliance Studio User Guide*.

To create batches for data movement and graph loading, follow these steps:

1. Perform one of the following:
  - Create and execute run executable. For more information, see [Appendix A, "Creating and Executing Run Executable"](#).
  - Alternatively, you can execute the `FCCM_Studio_SqoopJob.sh` command with the required parameters as follows:

---

**Note:** Before running the `FCCM_Studio_SqoopJob.sh` command, ensure that the Spark classpath does not contain `log4j jar` files. If it contains the `log4j jar` files, remove the following jars present in the `<PGX installed path>/pgx-3.1.2/lib` path.

---

```
./FCCM_Studio_SqoopJob.sh <Batch Name> <Batch ID> EXEC <FROM_DATE> <TO_DATE> SNAPSHOT_DT=<SNAPSHOT_Date>,DATAMOVEMENTCODE=ALL <SOURCE>
```

For example:

```
./FCCM_Studio_SqoopJob.sh A B C 20150618 20190618 SNAPSHOT_DT=20181219,DATAMOVEMENTCODE=ALL FCDM
```

Where,

- A is batchId
- 20150618 is FROM FIC MIS DATE
- 20180618 is FIC MIS DATE
- 20181219 is SNAPSHOT\_DT
- ALL is DATAMOVEMENT
- FCDM is source

## Executing Published Scenario Notebook

The published scenario notebook can be scheduled for execution with a set of threshold values as seemed required for generating alert or trends

To create batches for executing published scenario notebooks, follow these steps:

1. Perform one of the following:
  - Create and execute run executable. For more information, see [Appendix A, "Creating and Executing Run Executable"](#).
  - Alternatively, you can execute the following command with the required parameters as follows:

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

If the notebook is not a scenario notebook, then you can execute the notebook as follows:

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

For example, if you have used `ficmisdate` as a `paramkey1`, `lookbackperiod` as `paramkey2`, `value1` as `20-09-2018`, and `value2` as `30` in your notebook then `extraparams` should be written as follows:

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



---

---

# 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](#)

## 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](#)

## 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](#)

### 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 webserver) 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.

---

---

---

---

**Note:**

**In distributed mode: use the host which starts the webserver**

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 a http server.

---

---

### 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 -srcaias 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.

## 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.

## 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_password> -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 client.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 server_cert.pem -alias pgxserver
```

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

## 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.

## 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.

## 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.



---

## Configuring Interpreters

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


The various interpreters Studio are PGX, PGQL, GreenMarl, OFSAA Interpreter, OFSAA SQL Interpreter, Markdown and so on.

This chapter includes the following topics:

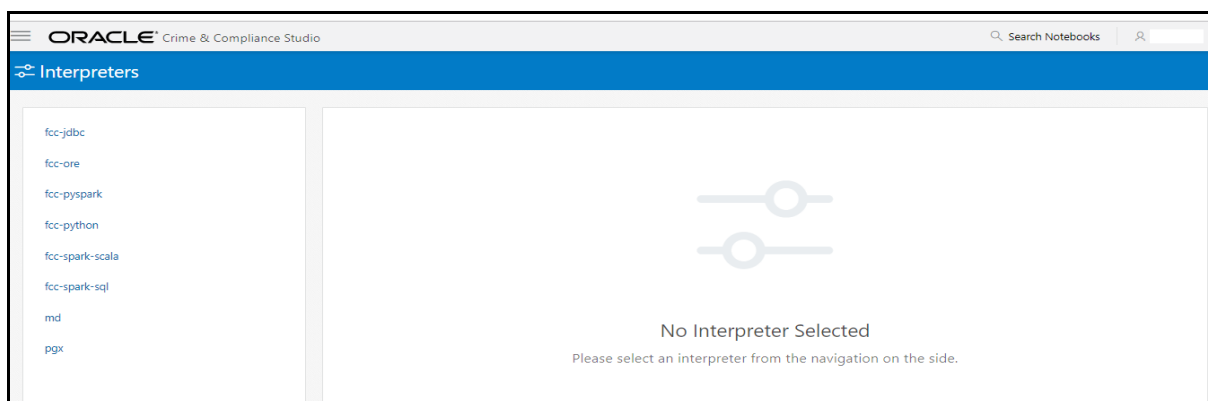
- [Accessing Interpreters](#)
- [Create New Interpreter Variant](#)
- [Configure Interpreters](#)

### Accessing Interpreters

To access interpreters, follow these steps:

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

**Figure 6–1** Interpreters Page



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 for the interpreter.

## Create New Interpreter Variant

In 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.

To create a new variant for an interpreter:

1. Navigate to the **Interpreters** page.
2. Click the interpreter for which you want to create a new variant 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. Enter the new schema details such as **default.url**, **default.user**, and **default.password**.
7. Click **Update**.
8. The database schema which you have created should be provided with grants that were provided to BD atomic schema.
9. Run the following SQL scripts after modifying the schema name with the newly created schema:

```
../OFS_FCCM_STUDIO/metaservice/model/SQLScripts/Atomic_Schema/FCC_
JRSDCN_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 new interpreter variant in notebook paragraphs, use the following format:  
`%fcc-jdbc.newVariant`  
`<Your SQL query>`
11. Configure required values for the various properties.
12. Click **Update**.

A new variant is created for the interpreter.

## Create 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 database schema which you have created should be provided with grants that were provided to the BD atomic schema.

For more information, see *Installer and Installation Prerequisites* section in the *Oracle Financial Services Crime and Compliance Studio Application Installation Guide*.

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

```
../OFS_FCCM_STUDIO/metaservice/model/SQLScripts/Atomic_Schema/FCC_
JRSDCN_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>
```

## Configure Interpreters

This section provides details of various interpreters in Studio and the configurations for each interpreter.

The various interpreters in Studio are as follows:

- [fcc-jdbc Interpreter](#)
- [fcc-ore Interpreter](#)

- [fcc-pyspark Interpreter](#)
- [fcc-python Interpreter](#)
- [fcc-spark-scala Interpreter](#)
- [fcc-spark-sql Interpreter](#)
- [md Interpreter](#)
- [pgx Interpreter](#)

## fcc-jdbc Interpreter

The configuration for the ofsaa-jdbc is given as follows:

**Table 6–1 fcc-jdbc interpreter Values**

Field	Description
default.url	Enter the ofsaa jdbc URL in this field. For example, jdbc:mysql://localhost:5554/world
default.user	Enter the name of the default user in this field. For example, root
default.password	Enter the default password.
pgx.baseUrl	Enter the pgx.baseUrl URL in this field. This is the location where the data is pushed. For example, http://##HOSTNAME##:7007
ofsaa.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.
ofsaa.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.
default.completer.ttlInSeconds	Enter the time to live sql completer in seconds.
default.driver	Enter the default ofsaa JDBC driver name. For example, com.mysql.jdbc.Driver
default.completer.schemaFilters	Enter comma separated schema filters to get metadata for completions
zeppelin.jdbc.precode	Enter the snippet of code that executes after the initialization of the interpreter.
default.splitQueries	This field indicates the presence of default split queries. Enter “true” or “false”.
common.max_count	Enter the maximum number of SQL result to display.
zeppelin.jdbc.auth.type	Enter the default jdbc authentication type.
zeppelin.jdbc.concurrent.use	Enter to enable or disable concurrent use of JDBC connections. Enter “true” or “false”.
zeppelin.jdbc.concurrent.max_connection	Enter the number of maximum connections allowed.
zeppelin.jdbc.keytab.location	Enter the keytab location.

**Table 6–1 fcc-jdbc interpreter Values**

Field	Description
zeppelin.jdbc.principal	Enter the principal name to load from the keytab.

## fcc-ore Interpreter

The configuration for the fcc-ore interpreter is given as follows:

**Table 6–2 fcc-ore Interpreter Values**

Field	Description
dummy.property.name	Enter any name
ore.host	Enter the hostname of the DB server where the fcc-ore interpreter wants to connect.
ore.port	Enter the port number of the DB server where the fcc-ore interpreter wants to connect.
ore.sid	Enter the SID of DB server where the fcc-ore interpreter wants to connect.
ore.user	Enter the schema name where the fcc-ore interpreter wants to connect.
ore.password	Enter the schema password where the fcc-ore interpreter wants to connect.
ore.service_name	Enter the Service Name of DB server where the fcc-ore interpreter wants to connect.
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.
ore.type	Enter the fcc-ore interpreter type as Oracle.
ore.all	Indicates all tables are synced to the fcc-ore interpreter. Enter the value as True.
ofsaa.metaservice.url	Enter the metaservice URL in this field. For example, <code>http://##HOSTNAME##:7045/metaservice</code> Here, <code>##HOSTNAME##</code> refers to the server name or IP where fcc-studio will be installed.
ofsaa.sessionservice.url	Enter the session service URL in this field. For example, <code>http://##HOSTNAME##:7047/sessionservice</code> Here, <code>##HOSTNAME##</code> refers to the server name or IP where fcc-studio will be installed.
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>
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>

**Table 6–2 fcc-ore Interpreter Values**

Field	Description
repo_cran	Indicates the CRAN URL from where R libraries are downloaded to install R packages. For example, <a href="https://cran.r-project.org/">https://cran.r-project.org/</a>
libpath	Indicates the custom library path from where R packages will be installed via Studio and will be added to R lib Path. Enter the path to be mentioned under the home directory where the studio is installed. For example, If you want the packages to be available under /home/user/library, and Studio is installed at /home/user/datastudio, then mention /library as the libpath.
rendering.row.limit	Indicates the number of rows to be shown in the fcc-ore interpreter output. For example, 1000
rendering.numeric.format	Indicates the Number of digits to round off. Foe example, %.2f
rendering.include.row.name	Indicates whether to include row names. For example, false
rserve.host	Indicates the Rserve host.
rserve.port	Indicates the Rserve port.
rserve.user	Indicates the Rserve username.
rserve.password	Indicates the Rserve password.
rserve.secure.login	Enter TRUE to enforce secure login.
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
rserve.try.wrap	Enter False.
rendering.knitr.image.width	Indicates the image width specification for ore output. For example, 60
rendering.knitr.options	Enter the Knitr output rendering option. For example, out.format = 'html', comment = NA, echo = FALSE, results = 'verbatim', message = F, warning = F, dpi = 300

## fcc-pyspark Interpreter

The configuration for the fcc-pyspark interpreter is given as follows:

**Table 6–3 fcc-pyspark Interpreter Values**

Field	Description
pgx.baseUrl	Enter the pgx.baseUrl URL in this field. This is the location where the data is pushed. For example, <a href="http://##HOSTNAME##:7007">http://##HOSTNAME##:7007</a>

**Table 6–3 fcc-pyspark Interpreter Values**

Field	Description
ofsaa.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.
ofsaa.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.
zeppelin.livy.url	Enter the Livy URL in this field. Livy is an interface between Data Studio and Spark. For example, http://##HOSTNAME##:8998
zeppelin.livy.session.create_timeout	Enter the Zeppelin session creation timeout in seconds.
livy.spark.driver.cores	Enter the number of driver cores to use for the driver process.
livy.spark.driver.memory	Enter the amount of memory to use for the driver process.
livy.spark.executor.instances	Enter the number of executors to launch for the current session.
livy.spark.executor.cores	Enter the number of executor cores to use for the driver process.
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.achedExecutorIdleTimeout	Enter the cached execution timeout in seconds.
livy.spark.dynamicAllocation.minExecutors	Enter the minimum number of required Dynamic Allocation executors.
livy.spark.dynamicAllocation.initialExecutors	Enter the initial Dynamic Allocation executors.
livy.spark.dynamicAllocation.maxExecutors	Enter the maximum number of required Dynamic Allocation executors.
zeppelin.livy.principal	Enter the principal name to load from the keytab.
zeppelin.livy.keytab	Enter the keytab location.
zeppelin.livy.pull_status.interval.millis	Enter the data pull interval in milliseconds.
livy.spark.jars.packages	Enter to add extra libraries to a livy interpreter.
zeppelin.livy.displayAppInfo	This field indicates whether the application information needs to be displayed or not. Enter “true” or “false”.
zeppelin.livy.spark.sql.maxResult	Enter the maximum number of results that need to be fetched.

## fcc-python Interpreter

The configuration for the python interpreter is given as follows:

**Table 6–4 fcc-python Interpreter Values**

Field	Description
zeppelin.python	Enter the Python installed path.
zeppelin.python.maxResult	Enter the maximum number of results that need to be fetched.

## fcc-spark-scala Interpreter

The configuration for the ofsaas interpreter is given as follows:

**Table 6–5 fcc-spark-scala Interpreter Values**

Field	Description
pgx.baseUrl	Enter the pgx.baseUrl URL in this field. This is the location where the data is pushed. For example, http://##HOSTNAME##:7007
ofsaas.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.
ofsaas.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.
zeppelin.livy.url	Enter the Livy URL in this field. Livy is an interface between Data Studio and Spark. For example, http://##HOSTNAME##:8998
zeppelin.livy.session.create_timeout	Enter the Zeppelin session creation timeout in seconds.
livy.spark.driver.cores	Enter the number of driver cores to use for the driver process.
livy.spark.driver.memory	Enter the amount of memory to use for the driver process.
livy.spark.executor.instances	Enter the number of executors to launch for the current session.
livy.spark.executor.cores	Enter the number of executor cores to use for the driver process.
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.cachedExecutorIdleTimeout	Enter the cached execution timeout in seconds.
livy.spark.dynamicAllocation.minExecutors	Enter the minimum number of required Dynamic Allocation executors.
livy.spark.dynamicAllocation.initialExecutors	Enter the initial Dynamic Allocation executors.
livy.spark.dynamicAllocation.maxExecutors	Enter the maximum number of required Dynamic Allocation executors.
zeppelin.livy.principal	Enter the principal name to load from the keytab.
zeppelin.livy.keytab	Enter the keytab location.



**Table 6–5 fcc-spark-scala Interpreter Values**

Field	Description
zeppelin.livy.pull_status.interval.millis	Enter the data pull interval in milliseconds.
livy.spark.jars.packages	Enter to add extra libraries to a livy interpreter.
zeppelin.livy.displayAppInfo	This field indicates whether the application information needs to be displayed or not. Enter “true” or “false”.
zeppelin.livy.spark.sql.maxResult	Enter the maximum number of results that need to be fetched.

## fcc-spark-sql Interpreter

The configuration for the ofsa-sql interpreter is given as follows:

**Table 6–6 fcc-spark-sql Interpreter Values**

Field	Description
pgx.baseUrl	Enter the pgx.baseUrl URL in this field. This is the location where the data is pushed. For example, http://##HOSTNAME##:7007
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.
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.
zeppelin.livy.url	Enter the Livy URL in this field. Livy is an interface between Data Studio and Spark. For example, http://##HOSTNAME##:8998
zeppelin.livy.session.create_timeout	Enter the Zeppelin session creation timeout in seconds.
livy.spark.driver.cores	Enter the number of driver cores to use for the driver process.
livy.spark.driver.memory	Enter the amount of memory to use for the driver process.
livy.spark.executor.instances	Enter the number of executors to launch for the current session.
livy.spark.executor.cores	Enter the number of executor cores to use for the driver process.
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.cachedExecutorIdleTimeout	Enter the cached execution timeout in seconds.
livy.spark.dynamicAllocation.minExecutors	Enter the minimum number of required Dynamic Allocation executors.
livy.spark.dynamicAllocation.initialExecutors	Enter the initial Dynamic Allocation executors.

**Table 6–6 fcc-spark-sql Interpreter Values**

Field	Description
livy.spark.dynamicAllocation.maxExecutors	Enter the maximum number of required Dynamic Allocation executors.
zeppelin.livy.principal	Enter the principal name to lead from the keytab.
zeppelin.livy.keytab	Enter the keytab location.
zeppelin.livy.pull_status.interval.millis	Enter the data pull interval in milliseconds.
livy.spark.jars.packages	Enter to add extra libraries to a livy interpreter.
zeppelin.livy.displayAppInfo	This field indicates whether the application information needs to be displayed or not. Enter “true” or “false”.
zeppelin.livy.spark.sql.maxResult	Enter the maximum number of results that need to be fetched.

## md Interpreter

The configuration for the md interpreter is given as follows:

**Table 6–7 md Interpreter Values**

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

## pgx Interpreter

For information on Parallel Graph Analytix (PGX), refer the following links:

<https://www.oracle.com/middleware/technologies/parallel-graph-analytix.html>

[https://docs.oracle.com/cd/E56133\\_01/latest/index.html](https://docs.oracle.com/cd/E56133_01/latest/index.html)

For information on Property Graph Query Language (PGQL), refer the following links:

<http://pgql-lang.org/>

The configuration for the pgx interpreter is given as follows:

**Table 6–8 pgx Interpreter Values**

Field	Description
pgx.timeout	Enter the pgx session creation timeout in seconds.
pgx.baseUrl	Enter the pgx.baseUrl URL in this field. This is the location where the data is pushed. For example, <a href="http://##HOSTNAME##:7007">http://##HOSTNAME##:7007</a>
pgx.trustStore	Enter the zeppelin.livy.url URL in this field. For example, <a href="http://##HOSTNAME##:8998">http://##HOSTNAME##:8998</a>
pgx.keyStore	Enter the Zeppelin session creation timeout in seconds.
pgx.password	Enter the pgx password.
pgx.accessToken	Enter the pgx access token.
pgx.prettyprint	Enter the pgx pretty print.
pgx.visualizePgqlResults	Enter the pgx visualize pgql results.

**Table 6–8** *pgx Interpreter Values*

<b>Field</b>	<b>Description</b>
pgx.maxResults	Enter the maximum number of results that need to be fetched.



## Configuring Data Sources for Graph

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 already 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 7-1** *fcc\_studio\_etl\_queries Table Details*

Column Name	Description
Type	Indicates the column name. Enter the value as NODE or EDGE.
DF_NAME	Indicates the name for the node or edge.
SOURCE	Indicates the source of the data. For example, FCDM or ICIJ
DATAFRAME	Indicates the properties of the node or edge. <b>Note:</b> Enter this value only if the data source is Hive and not CSV.
QUERY	<ul style="list-style-type: none"> <li>• If the source is Hive, provide the Hive query.</li> <li>• If the source is CSV, provide the query in the following format:  <pre>spark.read.format("csv").option("header", "true").option("mode", "DROPMALFORMED").load("##FILEPATH##").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> </li> </ul> For information on the query parameters, see <a href="#">Table 7-2, "Query Parameter Details"</a>

**Table 7-2** *Query Parameter Details*

Query Parameter	Description
<code>spark.read.format("csv")</code>	Indicates the input file format. For example, .csv.

**Table 7–2 Query Parameter Details**

Query Parameter	Description
option("header", "true")	Indicates the presence of header in the input file. <ul style="list-style-type: none"> <li>• True indicates that the header is available in the input file.</li> <li>• False indicates that the header is absent in the input file.</li> </ul>
load("Path").	Load indicates to load the data from the mentioned file path. Here, 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 A to B.
filter(col("A")==="Test1")	Indicates the "Where" filter condition, where 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 CSV, configure the file path in the fcc\_studio\_etl\_files table.
5. Enter the following details in the fcc\_studio\_etl\_files table to add file path:

**Table 7–3 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 seq ID for each file.
FILEPATH	Enter the path where CSV files are stored. <b>Note:</b> If one data frame has multiple CSV files, then make separate entries for all the files. For example, see <a href="#">Figure 7–1</a> .
FILEORDER	If data is to be imported from multiple files specify the order in which files should be read.

**Figure 7–1 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

6. Navigate to the <ETL\_Installation\_Path>/etl/conf/ path.

- 
7. Configure the `etl.properties` file to add a new property.

Follow these steps:

- [Adding a New Connector in the etl.properties File](#)
  - [Adding a New Property in the etl.properties File](#)
8. Redeploy the Studio application. For more information, see the *Redeploy Studio Application* chapter in the *OFS Crime and Compliance Studio Deployment Guide (Using Kubernetes)*.

## Adding a New Connector in the etl.properties File

To add a new connector, follow these steps:

1. Add a new connector label to the connectors field.

---

---

**Note:** Ensure that the labels are separated by a semicolon.

---

---

2. Provide a location for the node or edge files as follows:

- For node:

```
<connector label>.nodes.location / <connector label>.nodes.location.<serial number>
```

- For edge:

```
<connector label>.edges.location / <connector label>.edges.location.<serial number> keys
```

3. For each node or edge file provide list of optional properties:

- For node:

```
<connector label>.node.property.list / <connector label>.node.property.list.<serial number>
```

- For edge:

```
<connector label>.edge.property.list / <connector label>.edge.property.list.<serial number>
```

## Adding a New Property in the etl.properties File

To add a new property, follow these steps:

1. Add the property name to the appropriate list of optional properties, depending on whether it is a node or edge property.

The property is either `optional.node.property.list` or `optional.edge.property.list`.

---

---

**Note:**

- Ensure that the labels are separated by a semicolon.
  - Ensure that the property name is spelled exactly the same as it appears in the connector output.
  - No escaping is required in this field if the property name contains spaces.
- 
-

- 
2. Add default value to the following property depending on whether you are adding node or edge property:

- For node:

```
<property name>.node.property.default
```

- For edge:

```
<property name>.edge.property.default
```

---

---

**Note:** You must use a backslash to escape space in the property name. For example, `Further\ Information.node.property.default`.

---

---

3. Add the property name list of the connector provided properties, to connector(s) providing new property. See `<connector label>.node.property.list` or `<connector label>.edge.property.list`.
4. (Optional) If the property needs to be renamed, you can use the renaming functionality mentioned above.
5. (Optional) If the property is not of String type, you can use the data type functionality above to set the PGX-specific data type of the property.



---

## Creating and Executing Run Executable

To create and execute run executable, follow these steps:

1. Login 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.

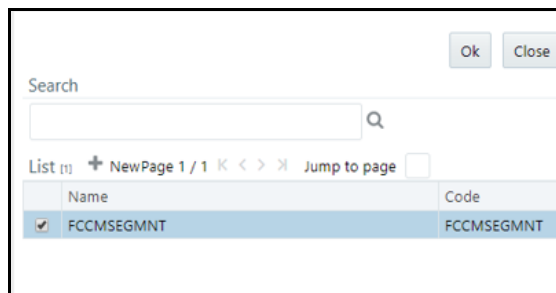
The **Run Definition** page is displayed.

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.

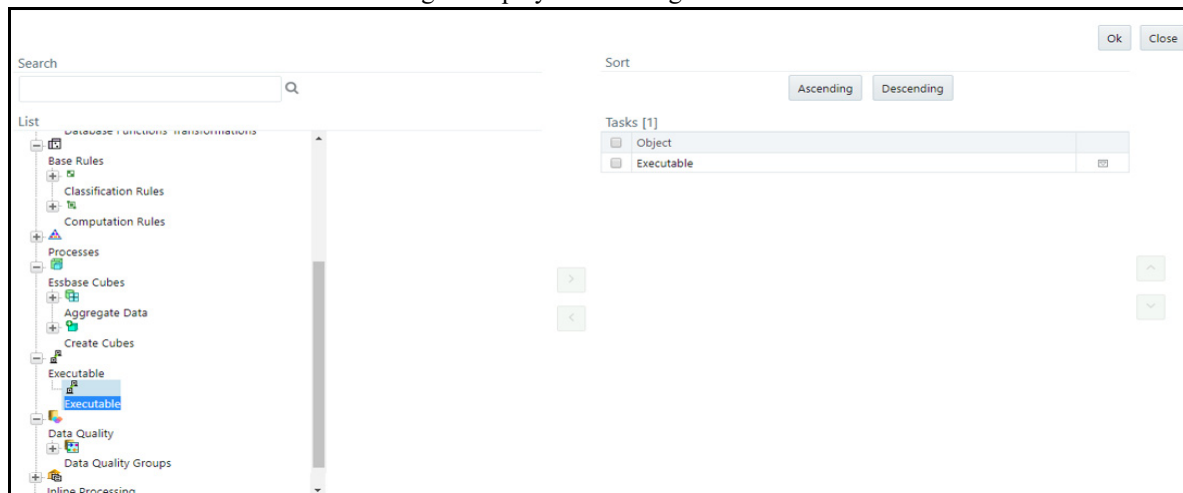


6. Select the folder that is to be linked to the run executable.
7. Click **OK**.
8. Enter the following details in the **Master Information** toolbar.

**Table A-1 Adding Run Definition**

Fields	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 Job Details 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:  

```
"FCCM_Studio_
NotebookExecution.sh", "notebookID", "outputParagraphID", "scenarioID", "th
resholdsetID", "extraparams"
```

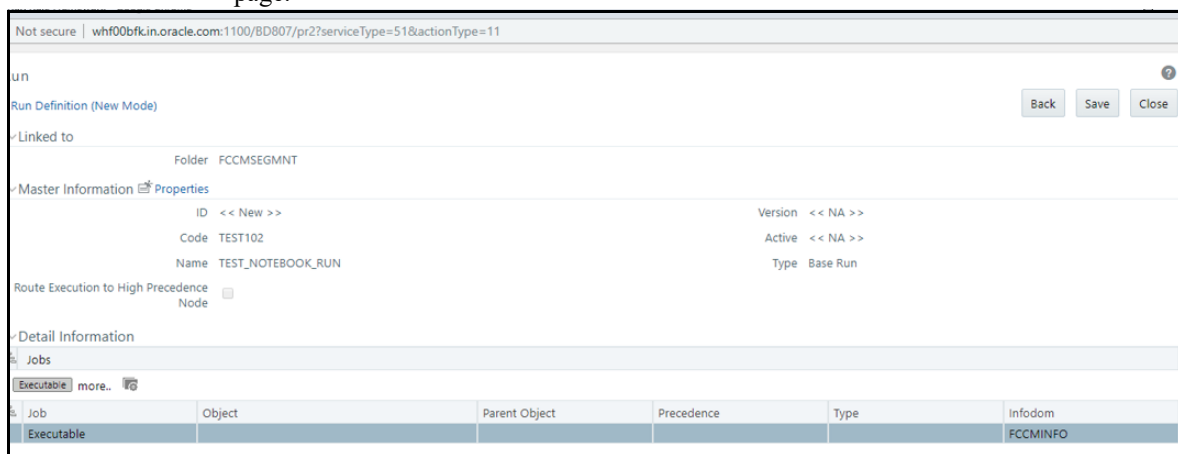
The following are the list of parameters for batch execution.

**Table A-2 List of parameters for batch execution**

No.	File Name	Parameters	Description
1	FCCM_Studio_DB_DataMove.sh		Data movement for Oracle database
	"FCCM_Studio_DB_DataMove.sh", "batchId", "ficmisdate", "userparams"		
2	FCCM_Studio_GraphLoad.sh		Graph loading for Oracle database
	"FCCM_Studio_GraphLoad.sh", "batchId", "ficmisdate", "userparams"		
3	FCCM_Studio_SqoopJob.sh		Data movement and graph load in Big Data Environment
	"FCCM_Studio_SqoopJob.sh", "batchId", "ficmisdate", "userparams"		
4	FCCM_Studio_NotebookExecution.sh		For batch execution of Scenario notebook
		notebookId	The ID of the required notebook
		outputParagraphId	It will always be "null".
		scenarioId	The ID of Scenario
		thresholdsetId	The ID of the threshold set with which notebook will run
		sessionId	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
	"FCCM_Studio_NotebookExecution.sh", "notebookID", "outputParagraphID", "scenarioID", "thresholdsetID", "extraparams"		

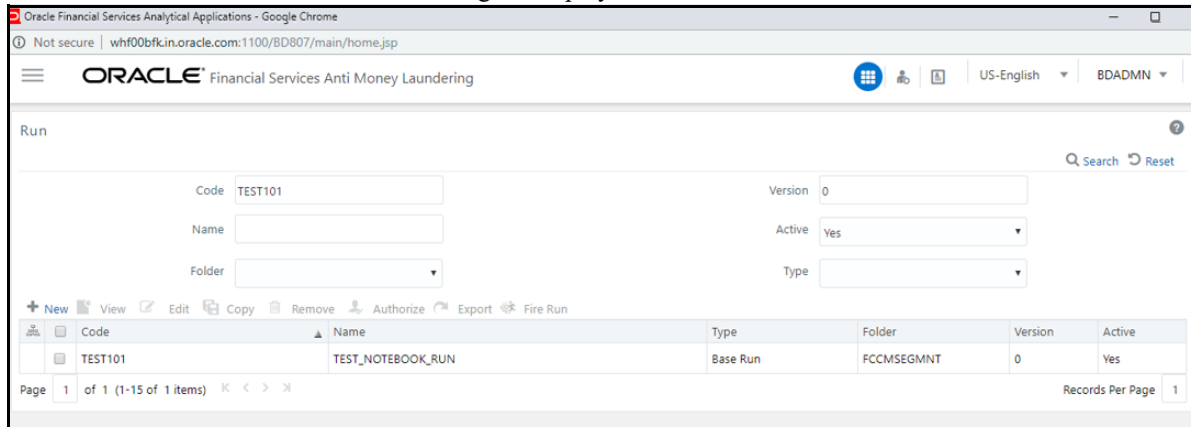
**14. Click OK.**

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



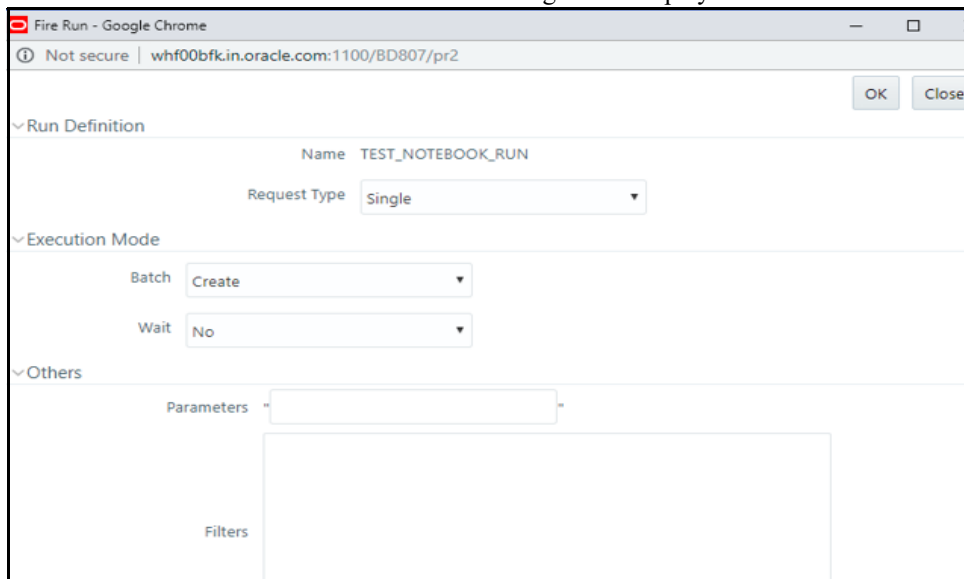
**15. Click Save.**

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 A-3 Adding Fire Run Details**

Fields	Description
Request Type	Select Request Type based on the following options: <ul style="list-style-type: none"> <li>• Single: If the batch has to be executed once.</li> <li>• Multiple: If the batch has to be executed multiple times at different intervals.</li> </ul>
Batch	Select Batch. It has the following options: <ul style="list-style-type: none"> <li>• Create</li> <li>• Create and Execute</li> </ul> From these options, select Create & Execute.

**Table A-3 Adding Fire Run Details**

<b>Fields</b>	<b>Description</b>
Wait	Select Wait. It has the following options: <ul style="list-style-type: none"><li>• Yes: This will execute the batch after a certain duration. Enter the duration as required.</li><li>• No: This will execute the batch immediately.</li></ul>
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.



This section includes the following topics:

- [ETL Service](#)
- [Data Forwarding Service](#)
- [Cross Language Name Matching Service](#)

## ETL Service

ETL enables you to prepare a Global Financial Graph from individual data sources. In addition, it identifies similar names and addresses across data sources and connects them with additional edges to simplify the analysis and exploration of relationships.

- [graph.config.template.json File Details](#)
- [etl.properties File Details](#)
- [Sample for etl.properties File](#)

## graph.config.template.json File Details

The Graph Configuration Template file, `graph.config.template.json` is present in the `<Studio_Installation_Path>/configmaps/etl/` path. The `graph.config.template.json` defines a template for the [PGX graph configuration file](#) that the ETL finally produces.

For example:

```
{
  "edge_props" : [
    %EDGE_PROPERTY_LIST%
  ],
  "vertex_props": [
    %NODE_PROPERTY_LIST%
  ],
  "format" : "csv",
  "separator" : ",",
  "vertex_id_type": "long",
  "edge_uris" : ["%SERVER_NAME%/edges.csv?token=%SERVER_SECRET%"],
  "vertex_uris" : ["%SERVER_NAME%/nodes.csv?token=%SERVER_SECRET%"],
```

```
"local_date_format": [ "M/d/yyyy", "yyyy-MM-dd" ]
}
```

The placeholders %EDGE\_PROPERTY\_LIST% and %NODE\_PROPERTY\_LIST% are replaced dynamically by the ETL with the edge and node properties and their types.

As long as the output graph configuration file of the ETL is also the input for the forwarding server, then the placeholders %SERVER\_NAME% and %SERVER\_SECRET% are replaced dynamically by the forwarding server. For more information, see [Data Forwarding Service](#).

## etl.properties File Details

The `etl.properties` file is present in the `<Studio_Installation_Path>/configmaps/etl/` path. The `etl.properties` file contains the following type of keys:

- Fixed set of properties with fixed names. For more information, see [Fixed Set of Properties](#)
- Dynamically named properties with names depending on other properties. The dynamically named properties have a variable prefix (derived from connector name or property name) and fixed suffix. For more information, see [Dynamically Named Properties](#)

### Fixed Set of Properties

**Table 7–4 Fixed Set of Properties in `etl.properties` file**

Interaction Variable Name	Significance
connectors	A semicolon-separated list of connector labels. Individual labels will be used later to prefix properties connected to the particular connector (table with the connector output).
<b>List of Properties</b>	
required.node.property.list	A semicolon-separated list of mandatory node properties that have to be present in every node table produced by any connector.
required.edge.property.list	A semicolon-separated list of mandatory edge properties that have to be present in every edge table produced by any connector.
optional.node.property.list	A semicolon-separated list of optional node properties that can be present in node table produced by zero or more connectors. <b>Note:</b> The property may not be produced by any of the connectors, in that case, it will be constant in all nodes. The value is specified by property, <code>&lt;property name&gt;.node.property.default</code> .
optional.edge.property.list	A semicolon-separated list of optional edge properties that can be present in edge table produced by zero or more connectors. <b>Note:</b> The property may not be produced by any of the connectors, in that case, it will be constant in all edges. The value is specified by property, <code>&lt;property name&gt;.edge.property.default</code> .
node.id.col	A string defining column name for Node ID. It has to be produced by all connectors.
edge.from.id.col	A string defining column name for From ID, one end of an edge. It has to be produced by all connectors.
edge.to.id.col	A string defining column name for To ID, the other end of an edge. It has to be produced by all connectors.



**Table 7–4 Fixed Set of Properties in etl.properties file**

Interaction Variable Name	Significance
<b>Graph Output</b>	
output.nodes.location	A location for node file. It can either be a local or hdfs path.
output.edges.location	A location for edge file. It can either be a local or hdfs path.
output.config.template.location	A location for the graph configuration template file (see <a href="#">graph.config.template.json File Details</a> ) to be used for generating the final graph configuration file. If deploying the ETL with Kubernetes, provide the value as follows:  file:///configurations/etl/graph.config.template.json
output.config.location	A location for the final graph configuration file. The location can be either local or hdfs path. If you do not want a graph configuration file written, provide the value as follows:  null://EMPTY.
<b>Entity Resolution Related Properties</b>	
output.name.index.condition	A WHERE condition for a SQL statement to identify which nodes are persons or companies and will be put into the name list file of the Cross Language Name Matching.  <b>Note:</b> This string is directly concatenated into a SQL statement. (if the property contains space or you want to be sure, use `` around column name).
output.name.index.nodes.location	Local or hdfs path to a name list file. It has to be the same as name.index.location property in Cross Language Name Matching property file.
output.name.index.edges.location	The value should be null://EMPTY. The edge file is not considered in this case.
output.name.index.config.location	The value should be null://EMPTY. The graph configuration file is not considered in this case.
output.address.index.condition	A WHERE condition for a SQL statement to identify which nodes are addresses and will be put in to address list file of the Cross Language Name Matching.  <b>Note:</b> This string is directly concatenated into a SQL statement. (if the property contains space or you want to be sure, use `` around column name).
output.address.index.nodes.location	Local or hdfs path to an address list file. The value should be the same as the address.index.location property in Cross Language Name Matching property file.
output.address.index.edges.location	The value should be null://EMPTY. The edge file is not considered in this case.
output.address.index.config.location	The value should be null://EMPTY. The graph configuration file is not considered in this case.
name.matching.value.column	When storing the name list file, which node property to store as the name in the file. It has to match the name.matching.value.column property in Cross Language Name Matching property file.

**Table 7-4 Fixed Set of Properties in etl.properties file**

<b>Interaction Variable Name</b>	<b>Significance</b>
name.matching.filter.column	When storing the name list file, which node property to store as a filter in the file. It has to match the name.matching.filter.column property in Cross Language Name Matching property file.
address.matching.value.column	When storing an address list file, which node property to store as an address in the file. It has to match address.matching.value.column property in Cross Language Name Matching property file.
address.matching.filter.column	When storing an address list file, which node property to store as a filter in the file. It has to match address.matching.filter.column property in Cross Language Name Matching property file.
<b>Similarity Edges Generation Related Properties</b>	
similarity.label.edge.col	An edge property name where to store a label for similarity edge (typically label property).
similarity.value.edge.col	An edge property name, where to store a similarity score.
name.match.property.to.measure	When matching names, the node property that contains person/company names.
name.match.max.matches	When matching names, how many similarity edges create at most.
name.match.min.score	The name similarity edges are created for names with similarity scores specified here or higher.
address.match.property.to.measure	When matching addresses, the node property that contains addresses.
address.match.max.matches	When matching names, how many similarity edges create at most.
address.match.min.score	The address similarity edges are created for names with similarity score specified here or higher.
name.match.similarity.index.query.condition	A WHERE condition for a SQL statement. When adding name similarity edges between persons/companies, which nodes will be used as a basis (index) for matching. <b>Note:</b> This string is directly concatenated into a SQL statement. (if the property contains space or you want to be sure, use `` around column name).
name.match.similarity.match.query.condition	A WHERE condition for a SQL statement. When adding name similarity edges, between persons/companies, these nodes will be matched against the basis (index). <b>Note:</b> This string is directly concatenated into a SQL statement. (if the property contains space or you want to be sure, use `` around column name).
address.match.similarity.index.query.condition	A WHERE condition for a SQL statement. When adding address similarity edges between addresses, these nodes will be used as a basis (index) for matching. <b>Note:</b> This string is directly concatenated into a SQL statement. (if the property contains space or you want to be sure, use `` around column name).

**Table 7–4 Fixed Set of Properties in etl.properties file**

Interaction Variable Name	Significance
address.match.similarity.match.query.condition	<p>A WHERE condition for a SQL statement. When adding address similarity edges, between addresses, these nodes will be matched against the basis (index).</p> <p><b>Note:</b> This string is directly concatenated into a SQL statement. (if the property contains space or you want to be sure, use `` around column name).</p>

**Dynamically Named Properties****Table 7–5 Dynamically Named Properties in etl.properties File**

Interaction Variable Name	Significance
<b>Graph Node/Edge Property related options</b>	
<optional property>.node.property.default	A default value for each node property defined in optional.node.property.list. If a connector does not supply the property, it will be filled automatically, using this default value.
<optional property>.edge.property.default	A default value for each edge property defined in optional.node.property.list. If a connector does not supply the property, it will be filled automatically, using this default value.
<property>.node.property.rename.to	A value to rename the node property defined in either required.node.property or optional.node.property.list in the final output.
<property>.edge.property.rename.to	A value to rename the edge property defined in either required.edge.property or optional.edge.property.list in the final output.
<property>.node.property.data.type	The PGX type to use for the node property when writing the graph configuration file. ('string' is the default).
<property>.edge.property.data.type	The PGX type to use for the edge property when writing the graph configuration file ('string' is the default).
<b>Connector Related Properties</b>	
<b>If a connector provides a single node/edge table</b>	
<connector label>.nodes.location	<p>A location of the node table. The value can be a local file, hdfs file or hive table.</p> <p><b>Note:</b> The &lt;connector label&gt; is one of the individual values from the connectors option.</p>
<connector label>.node.property.list	A semicolon-separated list of optional node properties provided by the connector (not listing the required properties).
<connector label>.edges.location	A location of the edge table. The value can be a local file, hdfs file or hive table.
<connector label>.edge.property.list	A semicolon-separated list of optional edge properties provided by the connector (not listing the required properties).
<b>If a connector provides multiple node/edge tables</b>	

**Table 7–5 Dynamically Named Properties in etl.properties File**

Interaction Variable Name	Significance
<connector label>.nodes.location.<serial number>	The location of one of the connector node tables. The value can be a local file, hdfs file or hive table. <b>Note:</b> The <connector label> is one of the individual values from the connectors option. The <serial number> is a sequence number, starting from 1.
<connector label>.node.property.list.<serial number>	A semicolon-separated list of optional node properties provided in corresponding table (not listing the required properties).
<connector label>.edges.location.<serial number>	The location of one of the connector edge tables. The value can be a local file, hdfs file or hive table.
<connector label>.edge.property.list.<serial number>	A semicolon-separated list of optional edge properties provided in corresponding table (not listing the required properties).

## Sample for etl.properties File

```
connectors=paradise;bahama;offshore;panama;fcdm
```

```
required.node.property.list=node_id;label;original_id
```

```
required.edge.property.list=from;to;label
```

```
optional.node.property.list=account_type;address;country;customer_
type;date;industry;is_pep;jurisdiction;name;reason;risk;rule_
name;source;status;List;Entity Type;Aliases;Category;External
Sources;Further Information;DB Customer Flag;Domicile Address
Country;Registered Address Country;Tax Country;Lexis Nexis Match Flag;DB
Client ID;Local Name;Naics Code;Tax ID
```

```
optional.edge.property.list=currency;date;edge_type;info;is_foreign_
transaction;is_path_through_transaction;relationship;source;weight
```

```
account_type.node.property.default=
```

```
address.node.property.default=
```

```
country.node.property.default=
```

```
customer_type.node.property.default=
```

```
date.node.property.default=1970-01-01
```

```
industry.node.property.default=
```

```
is_pep.node.property.default=
```

```
jurisdiction.node.property.default=
```

```
name.node.property.default=
```

```
reason.node.property.default=
```

```
risk.node.property.default=
```

```
rule_name.node.property.default=  
source.node.property.default=  
status.node.property.default=  
List.node.property.default=  
Entity\ Type.node.property.default=  
Aliases.node.property.default=  
Category.node.property.default=  
External\ Sources.node.property.default=  
Further\ Information.node.property.default=  
DB\ Customer\ Flag.node.property.default=  
Domicile\ Address\ Country.node.property.default=  
Registered\ Address\ Country.node.property.default=  
Tax\ Country.node.property.default=  
Lexis\ Nexis\ Match\ Flag.node.property.default=  
DB\ Client\ ID.node.property.default=  
Local\ Name.node.property.default=  
Naics\ Code.node.property.default=  
Tax\ ID.node.property.default=  
  
currency.edge.property.default=USD  
date.edge.property.default=1970-01-01  
edge_type.edge.property.default=  
info.edge.property.default=  
is_foreign_transaction.edge.property.default=false  
is_path_through_transaction.edge.property.default=false  
relationship.edge.property.default=  
source.edge.property.default=  
weight.edge.property.default=0.0  
  
account_type.node.property.rename.to=Account Type  
address.node.property.rename.to=Address  
country.node.property.rename.to=Country  
customer_type.node.property.rename.to=Customer Type  
date.node.property.rename.to=Date  
industry.node.property.rename.to=Industry  
is_pep.node.property.rename.to=Is PEP  
jurisdiction.node.property.rename.to=Jurisdiction
```

```
name.node.property.rename.to=Name
reason.node.property.rename.to=Reason
risk.node.property.rename.to=Risk
rule_name.node.property.rename.to=Rule Name
source.node.property.rename.to=Source
status.node.property.rename.to=Status

currency.edge.property.rename.to=Currency
date.edge.property.rename.to=Date
edge_type.edge.property.rename.to=Edge Type
info.edge.property.rename.to=Info
is_foreign_transaction.edge.property.rename.to=Is Foreign Transaction
is_path_through_transaction.edge.property.rename.to=Is Path Through
Transaction
relationship.edge.property.rename.to=Relationship
source.edge.property.rename.to=Source
weight.edge.property.rename.to=Weight

node.id.col=node_id
edge.from.id.col=from
edge.to.id.col=to

similarity.label.edge.col=label
similarity.value.edge.col=weight
name.match.property.to.measure=name
address.match.property.to.measure=name

fcdm.nodes.location.1=hive://etlnew.fcdm_customer
fcdm.nodes.location.2=hive://etlnew.fcdm_account
fcdm.nodes.location.3=hive://etlnew.fcdm_address
fcdm.nodes.location.4=hive://etlnew.fcdm_derived_entity
fcdm.nodes.location.5=hive://etlnew.fcdm_event
fcdm.nodes.location.6=hive://etlnew.fcdm_institution
fcdm.node.property.list.1=node_id;label;original_
id;name;risk;source;date;jurisdiction;industry;customer_type;is_pep
fcdm.node.property.list.2=node_id;label;original_
id;name;risk;source;date;jurisdiction;account_type
```

```
fcdm.node.property.list.3=node_id;label;original_
id;name;risk;source;date;jurisdiction
```

```
fcdm.node.property.list.4=node_id;label;original_
id;name;risk;source;date;jurisdiction;is_pep
```

```
fcdm.node.property.list.5=node_id;label;original_
id;name;risk;source;date;jurisdiction;status;reason;rule_name
```

```
fcdm.node.property.list.6=node_id;label;original_
id;name;risk;source;date;jurisdiction
```

```
fcdm.edges.location.1=hive://etlnew.fcdm_end_to_end_wire
```

```
fcdm.edges.location.2=hive://etlnew.fcdm_event_on
```

```
fcdm.edges.location.3=hive://etlnew.fcdm_has_account
```

```
fcdm.edges.location.4=hive://etlnew.fcdm_is_related_to
```

```
fcdm.edges.location.5=hive://etlnew.fcdm_send_wire_to
```

```
fcdm.edges.location.6=hive://etlnew.fcdm_transfer_wire_to
```

```
fcdm.edges.location.7=hive://etlnew.fcdm_receive_wire_from
```

```
fcdm.edge.property.list.1=from;to;label;info;weight;currency;is_foreign_
transaction;is_path_through_transaction
```

```
fcdm.edge.property.list.2=from;to;label;info;weight;currency;is_foreign_
transaction;is_path_through_transaction
```

```
fcdm.edge.property.list.3=from;to;label;info;weight;currency;is_foreign_
transaction;is_path_through_transaction
```

```
fcdm.edge.property.list.4=from;to;label;info;weight;currency;is_foreign_
transaction;is_path_through_transaction
```

```
fcdm.edge.property.list.5=from;to;label;info;weight;currency;is_foreign_
transaction;is_path_through_transaction;date
```

```
fcdm.edge.property.list.6=from;to;label;info;weight;currency;is_foreign_
transaction;is_path_through_transaction;date
```

```
fcdm.edge.property.list.7=from;to;label;info;weight;currency;is_foreign_
transaction;is_path_through_transaction;date
```

```
bahama.nodes.location.1=hive://etlnew.icij_bahama_external_address_insert
```

```
bahama.nodes.location.2=hive://etlnew.icij_bahama_external_entity_insert
```

```
bahama.node.property.list.1=node_id;original_
id;address;jurisdiction;country;source;label
```

```
bahama.node.property.list.2=node_id;original_
id;name;jurisdiction;country;source;label
```

```
bahama.edges.location.1=hive://etlnew.icij_bahama_edges_officer_of_insert
```

```
bahama.edges.location.2=hive://etlnew.icij_bahama_edges_similar_company_
as_insert
```

```
bahama.edges.location.3=hive://etlnew.icij_bahama_edges_registered_
address_insert
bahama.edges.location.4=hive://etlnew.icij_bahama_edges_same_address_as_
insert
bahama.edges.location.5=hive://etlnew.icij_bahama_edges_same_company_as_
insert
bahama.edges.location.6=hive://etlnew.icij_bahama_edges_same_intermediary_
as_insert
bahama.edges.location.7=hive://etlnew.icij_bahama_edges_same_name_as_
insert
bahama.edge.property.list.1=from;to;edge_type;source;label
bahama.edge.property.list.2=from;to;edge_type;source;label
bahama.edge.property.list.3=from;to;edge_type;source;label
bahama.edge.property.list.4=from;to;edge_type;source;label
bahama.edge.property.list.5=from;to;edge_type;source;label
bahama.edge.property.list.6=from;to;edge_type;source;label
bahama.edge.property.list.7=from;to;edge_type;source;label

panama.nodes.location.1=hive://etlnew.icij_panama_external_address_insert
panama.nodes.location.2=hive://etlnew.icij_panama_external_entity_insert
panama.node.property.list.1=node_id;original_
id;address;jurisdiction;country;source;label
panama.node.property.list.2=node_id;original_
id;name;jurisdiction;country;source;label

panama.edges.location.1=hive://etlnew.icij_panama_edges_intermediary_of_
insert
panama.edges.location.2=hive://etlnew.icij_panama_edges_officer_of_insert
panama.edges.location.3=hive://etlnew.icij_panama_edges_registered_
address_insert
panama.edge.property.list.1=from;to;edge_type;relationship;source;label
panama.edge.property.list.2=from;to;edge_type;relationship;source;label
panama.edge.property.list.3=from;to;edge_type;relationship;source;label

paradise.nodes.location.1=hive://etlnew.icij_paradise_external_address_
insert
paradise.nodes.location.2=hive://etlnew.icij_paradise_external_entity_
insert
paradise.node.property.list.1=node_id;original_
id;address;jurisdiction;country;source;label
```



```
paradise.node.property.list.2=node_id;original_
id;name;jurisdiction;country;source;label
```

```
paradise.edges.location.1=hive://etlnew.icij_paradise_edges_intermediary_
of_insert
```

```
paradise.edges.location.2=hive://etlnew.icij_paradise_edges_officer_of_
insert
```

```
paradise.edges.location.3=hive://etlnew.icij_paradise_edges_registered_
address_insert
```

```
paradise.edge.property.list.1=from;to;edge_type;relationship;source;label
```

```
paradise.edge.property.list.2=from;to;edge_type;relationship;source;label
```

```
paradise.edge.property.list.3=from;to;edge_type;relationship;source;label
```

```
offshore.nodes.location.1=hive://etlnew.icij_offshore_external_address_
insert
```

```
offshore.nodes.location.2=hive://etlnew.icij_offshore_external_entity_
insert
```

```
offshore.node.property.list.1=node_id;original_
id;address;jurisdiction;country;source;label
```

```
offshore.node.property.list.2=node_id;original_
id;name;jurisdiction;country;source;label
```

```
offshore.edges.location.1=hive://etlnew.icij_offshore_edges_intermediary_
of_insert
```

```
offshore.edges.location.2=hive://etlnew.icij_offshore_edges_officer_of_
insert
```

```
offshore.edges.location.3=hive://etlnew.icij_offshore_edges_registered_
address_insert
```

```
offshore.edges.location.4=hive://etlnew.icij_offshore_edges_underlying_
insert
```

```
offshore.edge.property.list.1=from;to;edge_type;relationship;source;label
```

```
offshore.edge.property.list.2=from;to;edge_type;relationship;source;label
```

```
offshore.edge.property.list.3=from;to;edge_type;relationship;source;label
```

```
offshore.edge.property.list.4=from;to;edge_type;relationship;source;label
```

```
name.match.similarity.index.query.condition=label = 'Customer' AND name !=
'' AND source = 'Internal'
```

```
name.match.similarity.match.query.condition=label = 'External Entity' AND
source != 'Internal'
```

```
address.match.similarity.index.query.condition=label = 'Address' AND name
!= '' AND source = 'Internal'
```

```
address.match.similarity.match.query.condition=label = 'External Address'
AND source != 'Internal'
```

```
name.matching.value.column=name
name.matching.filter.column=source
address.matching.value.column=address
address.matching.filter.column=source
```

```
output.nodes.location=hdfs:///user/ofsaa/ETL_FILES/global_graph_nodes.csv
output.edges.location=hdfs:///user/ofsaa/ETL_FILES/global_graph_edges.csv
output.config.location=hdfs:///user/ofsaa/ETL_FILES/config.json
```

```
output.name.index.condition=label='Customer'
output.name.index.nodes.location=hdfs:///user/ofsaa/ETL_FILES/name_
index.csv
output.name.index.edges.location=null://EMPTY
output.name.index.config.location=null://EMPTY
```

```
output.address.index.condition=label='Address'
output.address.index.nodes.location=hdfs:///user/ofsaa/ETL_FILES/adress_
index.csv
output.address.index.edges.location=null://EMPTY
output.address.index.config.location=null://EMPTY
```

## Data Forwarding Service

Data Forwarding service provides support for fast and reliable loading of Global Financial Graph from the Big Data environment to FCC Studio.

- [forwarderServer.properties File Details](#)
- [Graph Configuration File Tokens](#)
- [Sample for forwarderServer.properties File](#)

### forwarderServer.properties File Details

The `forwarderServer.properties` file is present in the `<Studio_Installation_Path>/configmaps/dataforwardservice/ path`. The `forwarderServer.properties` file details are described in the following table:

**Table 7–6** *forwarderServer.properties Parameters*

Interaction Variable Name	Significance
listen.port	The port on which the server listens.

**Table 7–6 forwarderServer.properties Parameters**

Interaction Variable Name	Significance
listen.hostname	The IP address or hostname the server is bound to.
allowed.hostnames	A comma-separated list of hostnames that are allowed to communicate with the server or to allow communication with any computer.
secret.token	A random string value that a client has to send as part of the URL. The value can be anything. It is a simple security measure (preshared secret).
forwarder.server.hostname	A hostname of the server. This string will be filled in into PGX graph configuration file.
nodes.file.hdfs.path	HDFS path to the file with nodes.
edges.file.hdfs.path	HDFS path to the file with edges.
config.json.hdfs.path	HDFS path to the graph configuration template file.
buffer.size	Size of the buffer.
ssl.enabled	Flag whether communication will be SSL encrypted.
keystore.file.path	If SSL encryption is enabled, provide a path to the keystone file. Else ignore this parameter.
keystore.password	If SSL encryption is enabled, provide a password to the keystone file. Else ignore this parameter.
trusted.store.file.path	If SSL encryption is enabled, provide a path to the keystone file. Else ignore this parameter.
trusted.store.password	If SSL encryption is enabled, provide a password to the keystone file. Else ignore this parameter.

## Graph Configuration File Tokens

The configuration file may have the following placeholders which will be filled in by server in order to allow correct loading of the data. The placeholders specify the location of the edge and node files:

```
"edge_uris" : ["%SERVER_NAME%/edges.csv?token=%SERVER_SECRET%"],
```

```
"vertex_uris" : ["%SERVER_NAME%/nodes.csv?token=%SERVER_SECRET%"],
```

The `%SERVER_NAME%` is replaced by the value specified by `forwarder.server.hostname` and `%SERVER_SECRET%` is preshared secret, as specified by `secret.token`.

The above placeholders already exist in the recommended Graph Configuration Template, see example in [graph.config.template.json File Details](#). This makes it convenient when the ETL produces the graph configuration file at the same location as where the Data Forwarding Service expects it.

## Sample for forwarderServer.properties File

```
listen.port=7022
listen.hostname=0.0.0.0
allowed.hostnames=*
secret.token=top_secret
```

```

forwarder.server.hostname=localhost
nodes.file.hdfs.path=hdfs:///user/ofsa/ETL_FILES/global_graph_nodes.csv
edges.file.hdfs.path=hdfs:///user/ofsa/ETL_FILES/global_graph_edges.csv
config.json.local.path=hdfs:///user/ofsa/ETL_FILES/config.template.json
buffer.size=1048576
ssl.enabled=false
keystore.file.path=/configurations/certificates/key.store.jks
keystore.password=password
trusted.store.file.path=/configurations/certificates/trusted.store.jks
trusted.store.password=password
    
```

## Cross Language Name Matching Service

Cross Language Name Matching service matches arbitrary personal and company names provided by an analyst with personal and company names in Global Financial Graph.

- [NameMatchingLocations.properties File Details](#)
- [Sample for NameMatchingLocations.properties File](#)

## NameMatchingLocations.properties File Details

The NameMatchingLocations.properties file is present in the <Studio\_Installation\_Path>/configmaps/crosslangnamematch/ path. The NameMatchingLocations.properties file details is explained in the following table:

**Table 7-7 NameMatchingLocations.properties Parameters**

Interaction Variable Name	Significance
listen.port	The port on which the server listens
listen.hostname	The IP address or hostname to which the server is bound to.
allowed.hostnames	A comma-separated list of hostnames that are allowed to communicate with the server or to allow communication with any computer.
name.index.location	A local filesystem or HDFS path to the file containing a list of names (and node IDs).
address.index.location	A local filesystem or HDFS path to the file containing a list of addresses (and node IDs).
node.id.col	The column name in the name and address list files containing node IDs.
name.matching.value.column	The column name in the name list file containing names to be indexed
name.matching.filter.column	The column name in the name list file containing values for source filter. This column can be used to limit the returned names to a particular data source (set of data sources).
address.matching.value.column	The column name in the address list file containing addresses to be indexed.

**Table 7-7 NameMatchingLocations.properties Parameters**

Interaction Variable Name	Significance
address.matching.filter.column	The column name in the address list file containing values for source filter. This column can be used to limit the returned address to a particular data source (set of data sources).
max.matches.allowed	A maximal number of results returned from the Name Matching, regardless of the input parameter. It is a protection of the server, so it is not asked to send back a million results.
max.matches.default	The default number of results returned from the Name Matching when no value is specified in the request.
min.score.allowed	Minimal similarity score allowed. The similarity score of results will always be higher than this value, regardless of the input parameter. It is a protection of the name matching server, so it will not have to examine all the string looking for matches above 0.01.
min.score.name.default	Similarity score to be used when matching names if the parameter is missing in the request.
min.score.address.default	Similarity score to be used when matching addresses if the parameter is missing in the request.
input.weight.name.default	Input weight to be used when matching names if the parameter is missing in the request. Input weight influences how much features missing in the input influence the matching score.
input.weight.address.default	Input weight to be used when matching addresses if the parameter is missing in the request.
ssl.enabled	Flag to indicate whether communication will be SSL encrypted.
keystore.file.path	If SSL encryption is enabled, provide a path to the keystone file. Else ignore this parameter.
keystore.password	If SSL encryption is enabled, provide a password to the keystone file. Else ignore this parameter.
trusted.store.file.path	If SSL encryption is enabled, provide a path to the keystone file. Else ignore this parameter.
trusted.store.password	If SSL encryption is enabled, provide a password to the keystone file. Else ignore this parameter.

## Sample for NameMatchingLocations.properties File

```
listen.port=7023
listen.hostname=0.0.0.0
allowed.hostnames=*
name.index.location=hdfs:///user/ofsa/ETL_FILES/name_index.csv
address.index.location=hdfs:///user/ofsa/ETL_FILES/adress_index.csv
node.id.col=node_id
name.matching.value.column=name
name.matching.filter.column=source
address.matching.value.column=address
```

```
address.matching.filter.column=sources
ssl.enabled=false
keystore.file.path=/configurations/certificates/key.store.jks
keystore.password=password
trusted.store.file.path=/configurations/certificates/trusted.store.jks
trusted.store.password=password
```