

Oracle Financial Services Crime and Compliance Studio

Administration and Configuration Guide

Release 8.0.8.2.0

August 2022

E91246-01

ORACLE
Financial Services

OFS Crime and Compliance Studio Administration and Configuration Guide

Copyright © 2022 Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

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

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

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

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

Document Control

Version Number	Revision Date	Change Log
4	August 2022	<p>Updated notebookID details in the following section:</p> <ul style="list-style-type: none"> • Execute Published Scenario Notebook for FCC Studio with Non-OFSAA • Execute Published Non-Scenario Notebook for FCC Studio with Non-OFSAA <p>Updated the BD and ECM document reference links in the Access FCC Studio Using FCCRealm</p>
1.3	June 2022	Removed FAQ section. All are irrelevant queries.
1.2	January 2021	<p>Added the following sections:</p> <ul style="list-style-type: none"> • Similarity Edge Generation Job for Infer API (ML Name Boosted Matching Method) • Similarity Edge Generation Job for Training API • Enable Synonym and Stopword with the Elastic Search Service • Access FCC Studio with MMG <p>Updated the following sections:</p> <ul style="list-style-type: none"> • Create Rules in a Ruleset • Scoring Method
1.1	December 2020	<p>Added the following sections:</p> <ul style="list-style-type: none"> • Using FCC Studio REST API • List of Permissions
1.0	September 2020	Revamp

Table of Contents

1	Preface	9
1.1	Audience	9
1.2	Using Oracle Applications	9
1.2.1	<i>Help</i>	9
1.2.2	<i>Watch Video</i>	9
1.2.3	<i>Additional Resources</i>	9
1.2.4	<i>Conventions</i>	10
1.3	Contacting Oracle.....	10
1.3.1	<i>Access to Oracle Support</i>	10
1.3.2	<i>Comments and Suggestions</i>	10
2	About FCC Studio Administration.....	11
2.1	Administration Overview.....	11
2.1.1	<i>Key Concepts</i>	13
3	User Access and Permissioning Management.....	15
3.1	Access FCC Studio Using FCCRealm	16
3.2	Access FCC Studio Using SAMLRealm.....	17
3.3	Access FCC Studio Using IDCSRealm.....	18
3.4	Access FCC Studio with MMG	18
3.5	Manage Permissions	19
3.6	Work with Permissions	19
3.6.1	<i>Users</i>	20
3.6.2	<i>Groups</i>	20
3.6.3	<i>Roles</i>	21
3.6.4	<i>Permission Templates</i>	21
3.6.5	<i>Create New Groups, Roles, and Permission Templates</i>	22
3.7	List of Permissions.....	23
4	Interpreter Configuration and Connectivity	28
4.1	Configure Interpreters	29
4.1.1	<i>fcc-jdbc Interpreter</i>	32

4.1.2	<i>fcc-ore interpreter</i>	34
4.1.3	<i>fcc-pyspark Interpreter</i>	37
4.1.4	<i>fcc-python Interpreter</i>	39
4.1.5	<i>fcc-spark-scala Interpreter</i>	41
4.1.6	<i>fcc-spark-sql Interpreter</i>	43
4.1.7	<i>jdbc Interpreter</i>	45
4.1.8	<i>md Interpreter</i>	47
4.1.9	<i>pgql Interpreter</i>	48
4.1.10	<i>pgx-algorithm Interpreter</i>	48
4.1.11	<i>pgx-java Interpreter</i>	49
4.1.12	<i>pyspark Interpreter</i>	50
4.1.13	<i>spark Interpreter</i>	50
4.2	Link Credentials.....	52
4.3	Create a Credential	53
4.4	Create an Interpreter Variant	55
4.5	Enable Second Spark or PySpark interpreter	56
4.6	Modify the Python Docker Images for the Python Interpreter	56
4.6.1	<i>Prerequisite to Build a Python Interpreter Docker Image</i>	57
4.6.2	<i>Build and Push an Image</i>	59
5	Prepare the Batches	61
5.1	Prepare Batches for FCCRealm.....	61
5.2	Prepare Batches for FCCSAML Realm	61
6	Execute Published Notebook	63
6.1	Execute Published Scenario Notebook for FCC Studio with Non-OFSAA.....	63
6.2	Execute Published Non-Scenario Notebook for FCC Studio with Non-OFSAA	64
7	Enable Synonym and Stopword with the Elastic Search Service	66
8	Configure ETL	67
8.1	Understand ETL.....	67
8.1.1	<i>Data Source</i>	67
8.1.2	<i>Rulesets</i>	67

8.1.3	<i>ElasticSearch</i>	68
8.1.4	<i>Indices</i>	68
8.1.5	<i>PGX</i>	68
8.1.6	<i>ETL and its Workflow</i>	69
8.1.7	<i>Jobs</i>	69
8.1.8	<i>Graph Model</i>	72
8.2	Configure Rulesets.....	72
8.2.1	<i>Use Rulesets</i>	73
8.2.2	<i>Create Rulesets</i>	74
8.2.3	<i>Create Rules in a Ruleset</i>	76
8.2.4	<i>Scoring Method</i>	78
8.3	Configure a Data Source	82
8.3.1	<i>Configure Spark Query Parameters</i>	84
8.4	Configure Graph.....	85
8.4.1	<i>Configure Attributes in Graph</i>	86
8.4.2	<i>Configure Extra Empty Nodes and Edges Providers</i>	87
8.4.3	<i>Additional Configuration (Local Date Format)</i>	88
8.5	Apply Graph Fine-Grained Access Control	89
9	Execute ETL	91
9.1	Prepare the Batches	91
9.1.1	<i>Prepare Batches for FCCRealm</i>	91
9.1.2	<i>Prepare Batches for FCCSAML Realm</i>	92
9.2	Perform the Batches.....	92
9.2.1	<i>Run ETL</i>	93
9.2.2	<i>Sqoop Job</i>	93
9.2.3	<i>Connector Job</i>	94
9.2.4	<i>Graph Job</i>	95
9.2.5	<i>Similarity Edge Generation Job</i>	96
9.2.6	<i>Similarity Edge Generation Job for Infer API (ML Name Boosted Matching Method)</i>	96
9.2.7	<i>Similarity Edge Generation Job for Training API</i>	96
9.3	Verify Batch Execution.....	97
9.3.1	<i>Verify Sqoop Job</i>	97

9.3.2	<i>Verify Connector Job</i>	98
9.3.3	<i>Verify Graph Job</i>	99
9.3.4	<i>Verify Similarity Edge Generation Job</i>	99
9.3.5	<i>Verify Oracle Schema Tables</i>	99
9.3.6	<i>Clean up for ETL</i>	99
10	Monitor Tasks	101
10.1	View Tasks Using Status.....	101
10.2	View Tasks Using Time of Creation	102
10.2.1	<i>View Tasks Using Names of Notebook</i>	103
11	Restart Services	105
11.1	Stop and Start the FCC Studio Services	105
11.2	Stop and Start the PGX Service.....	105
12	Configure Quantifind	106
13	Using FCC Studio REST API	109
13.1	Management of User Session	109
13.1.1	<i>POST sessions login FCCMRealm</i>	109
13.1.2	<i>POST sessions login for SAMLRealm</i>	110
13.2	Management of Roles.....	111
13.2.1	<i>GET roles</i>	111
13.2.2	<i>POST roles</i>	113
13.2.3	<i>GET roleId</i>	115
13.2.4	<i>PUT roleId</i>	116
13.2.5	<i>DELETE roleId</i>	117
14	Appendix - Create and Execute a Run Executable	118
15	Appendix Example of ETL	124

1 Preface

This guide provides information related to end-user tasks in the Oracle Financial Services (OFS) Crime and Compliance Studio (FCC Studio) application.


1.1 Audience

This guide is intended for Administrators and the basic knowledge of the following is recommended:

- UNIX commands
- Database concepts
- Big Data
- Python
- Scala
- Spark
- Oracle R
- SQL
- Groovy
- Markdown

1.2 Using Oracle Applications

1.2.1 Help

Use help icons Help Icon  to access help in the application. If you don't see any help icons on your page, click your user image or name in the global header and select Show Help Icons. Not all pages have help icons. You can also access the [Oracle Help Center](#) to find guides and videos.

1.2.2 Watch Video



Watch: This video tutorial shows you how to find and use help.

You can also read about it instead.

1.2.3 Additional Resources

- Community: Use [Oracle Cloud Customer Connect](#) to get information from experts at Oracle, the partner community, and other users.
- Training: Take courses on Oracle Cloud from [Oracle University](#).

1.2.4 Conventions

The following table explains the text conventions used in this guide.

Convention	Description
<i>Italics</i>	<ul style="list-style-type: none"> Names of books, chapters, and sections as references Emphasis
Bold	<ul style="list-style-type: none"> The object of an action (menu names, field names, options, button names) in step-by-step procedures Commands typed at a prompt User input
Monospace	<ul style="list-style-type: none"> Directories and subdirectories File names and extensions Process names Code sample, including keywords and variables within a text and as separate paragraphs, and user-defined program elements within a text
<Variable>	Substitute input value

1.3 Contacting Oracle

1.3.1 Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit [My Oracle Support](#) or visit [Accessible Oracle Support](#) if you are hearing impaired.

1.3.2 Comments and Suggestions

Please give us feedback about Oracle Applications Help and guides! You can send an e-mail to: [My Oracle Support \(MOS\)](#).

2 About FCC Studio Administration

The Financial Crime and Compliance (FCC) Studio uses the latest technology to harness the power of Graph Analytics to give Financial Institutions the ability to effectively monitor anti-money laundering and anti-fraud programs in financial institutions.

By using the FCC Studio, financial institutions can deploy advanced machine learning algorithms and artificial intelligence to discover unknown criminals, streamline compliance operations, and have unprecedented control over their Financial Crime program.

FCC Studio helps financial institutions to overcome the following challenges:

- Fraud programs in financial institutions.
- Identify and adapt to the changing patterns of financial crime.
- Ability to discover new and emerging criminal behavioral patterns.
- The ability of institutions to gain insights on the financial crime data with new and emerging financial crime patterns and trends.

The following are the key configurable features:

- Create users and roles to access FCC Studio to access through Realms
- Assign roles and group with required permissions
- The ability to customized and create interpreter variants to provide or restrict access to users
- Modify ready-to-use Python packages and versions
- Customize rulesets to generate similarity edges by configuring source and target entities
- Apply Graph Fine-Grained Access Control to redact the sensitive data in the Graphs
- Move source data to the PGX server using connector jobs to create graphs in FCDM and ICIJ workflows
- Monitor tasks that are performed by the logged-in users
- Offers ready to use extract, transform, load (ETL) operations.
- Entity resolution based on configurable rules.

2.1 Administration Overview

This section provides an overview of administration activities performed by an Administration after the installation of the FCC Studio application.

The following are the key configuration activities performed by an Administrator in FCC Studio:

- [Authenticate Users and Roles](#): To access the application, users must be authenticated. In FCC studio, users and roles are authenticated based on Realms, for example, FCCRealm, SAMLRealm, and IDCSRealm. These Realms use Identity

Management systems to authenticate users. FCCRealm - uses Oracle Financial Services Analytical Applications Infrastructure (OFSAI), SAMLRealm uses an identity provider (IDP), and IDCSRealm uses Oracle Identity Cloud Service (IDCS).

- **Authorize Users and Roles:** After authentication of users and roles, they must be authorized to use the application. The FCC Studio offers a rich permission system, users are mapped to the permissions to use application.
- **Configure Interpreters:** Interpreters are used to execute codes in different languages. They are plug-ins, which enable users to use a specific language to process data in the database. The FCC Studio provides ready-to-use interpreters, for example, jdbc-interpreter, python interpreter, and so on. In FCC Studio, you can either use a default interpreter variant or create a new variant for an interpreter to provide access to the database for different users. To enable secure data access, Interpreters are linked using credentials (a wallet and a password). Interpreters are configured based on usage.
- **Create Ruleset for Similarity Edges:** Ruleset facilitates to identify the similarity between two entities (customer, account, and so on) and derive a match. A Ruleset is a set of rules that are applied to the source and target entities, compares the attributes of the entities to derive a match.
- **Configure Data Source:** The data source configuration allows you to view the newly added edges or nodes in the graph. Define the source of the data, specify the order in which the files must be read, and so on.
- **Configure Graph:** The FCC Studio provides an intuitive way for creating graphs used in notebooks, where you can load graphs from external sources or create custom graphs. Using PGX, you can load multiple graphs into a notebook and create PGQL queries against different graphs. The result obtained from running a paragraph in a notebook can be used as an input to other paragraphs in the notebook. The results of analytics algorithms are stored as transient properties of nodes and edges in the graph. Pattern matching can then be used against these properties.
- **Apply Graph Fine-Grained Access Control and Redaction:** The Graph Fine-Grained Access Control and Redaction changes are applied to the FCC Studio to redact the sensitive data in the Graphs. With a role-based access control approach, you can restrict access at any level of granularity.
- **Batches:** Batches are prepared to execute the ETL operations. Batches enable you to move data from Oracle Database or Big Data to FCC Studio, load graphs, and run notebooks. Batches are prepared based on the Realms you are using.
- **Perform Batches:** These batches contain Sqoop Job, Connector Job, Graph Job, and Similarity Edge Generation Job in OFSAI. You can also execute ETL operations by running the scripts without configuring the batches.
- **Verify Batches Execution:** Verify the status of all tasks at the end of the batch execution. You can verify both the overall status of the batch and individual task status.
- **Execute Published Notebook:** The published JDBC notebooks are scheduled for execution with a set of threshold values required for generating alerts or trends.

- **Monitor Tasks:** Tasks are created when notebooks or paragraphs are executed by the end-user. It is important to know the status of the execution whether the tasks are created, rejected, canceled, and so on. The Tasks page allows you to view the status of the task and associated notebooks, paragraphs, interpreters, and so on. By default, all the tasks are listed on the Task page. You can view the specific task using filters such as status of the task, date of creation, name of the notebook.

2.1.1 Key Concepts

This section provides insight into the following key concepts:

- **Interpreter:** An interpreter is a program that directly executes instructions written in a programming or scripting language, without requiring them previously to be compiled into a machine language program. They are plug-ins, which enable users to use a specific language to process data in the back-end. Examples of Interpreters are, jdbc-interpreter, spark-interpreters, python-interpreters, and so on. Interpreters allow you to define customized drivers, URLs, passwords, connections, SQL result to display, and so on.
- **Zeppelin Interpreter:** A plug-in that enables Zeppelin users to use a specific language or data-processing-backend. For example, to use the Scala code in Zeppelin, you need a %spark interpreter.
- **Zeppelin:** Interactive browser-based notebooks that enable data engineers, data analysts, and data scientists to be more productive by developing, organizing, executing, and sharing data code and visualizing results without referring to the command line or requiring the cluster details. Notebooks allow these users not only allow to execute but to interactively work with long workflows.
- **Data discovery, exploration, reporting, and visualization** are key components of the data science workflow. Zeppelin provides a “Modern Data Science Studio” that supports ready-to-use Spark and Hive. Zeppelin supports multiple language backends which has support for a growing ecosystem of data sources. Zeppelin’s notebooks provide interactive snippet-at-time experience to data scientists. You can see a collection of Zeppelin notebooks in the Hortonworks Gallery.
- **PGQL:** A graph query language built on top of SQL, bringing graph pattern matching capabilities to existing SQL users and to new users who are interested in graph technology but who do not have an SQL background.
- **Keytab File:** A Keytab is a file containing pairs of Kerberos principles and encrypted keys (which are derived from the Kerberos password). You can use a keytab file to authenticate to various remote systems using Kerberos without entering a password. However, when you change your Kerberos password, you will need to recreate all your keytabs files. They are commonly used to allow scripts to automatically authenticate using Kerberos, without requiring human interaction or access to the password stored in a plain-text file. The script is then able to use the acquired credentials to access files stored on a remote system.
- **Markdown (md):** A plain text formatting syntax designed so that it can be converted to HTML. Use this section to configure the markdown parse type.

- **Parallel Graph AnalytiX (PGX):** Graph analysis lets you reveal latent information that is not directly apparent from fields in your data, but is encoded as direct and indirect relationships - metadata - between elements of your data. This connectivity-related information is not obvious to the naked eye but can have tremendous value when uncovered. PGX is a toolkit for graph analysis, supporting both efficient graph algorithms and fast SQL-like graph pattern matching queries.
- **PySpark:** A Python API is written in Python to support Spark. Spark is a distributed framework that can handle Big Data analysis. Spark is a computational engine that works with huge sets of data by processing them in parallel and batch systems.
- **Spark:** A fast and general-purpose cluster computing system. It provides high-level APIs in Java, Scala, Python, and R. Spark is an optimized engine that supports general execution graphs.
- **Oracle Wallet:** Oracle Wallet is a file that stores database authentication and signing credentials. It allows users to securely access databases without providing credentials to third-party software, and easily connect to Oracle products.
- **Sqoop Job:** A Sqoop is a tool designed for efficiently transferring bulk data between Hadoop and structured datastores such as relational databases. Sqoop job creates and saves the import and export commands. It specifies parameters to identify and recall the saved job. This re-calling or re-executing is used in the incremental import, which can import the updated rows from the RDBMS table to Hadoop Distributed File System (HDFS).
- **Elasticsearch:** An Elastic search is a distributed search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured.

3 User Access and Permissioning Management

FCC Studio uses a realm based on unique authentication and authorization for its users. Realm indicates the functional grouping of database schemas and roles that must be secured for an application. Realms protect data from access through system privileges and do not provide additional privileges to its owner or participants.

Realm based authorization establishes a set of database accounts and roles that can manage or access objects protected in realms and are authorized to use its system privileges. It provides a run-time mechanism to check logically if a user's command is allowed to access objects specified in the command and to proceed with its execution.

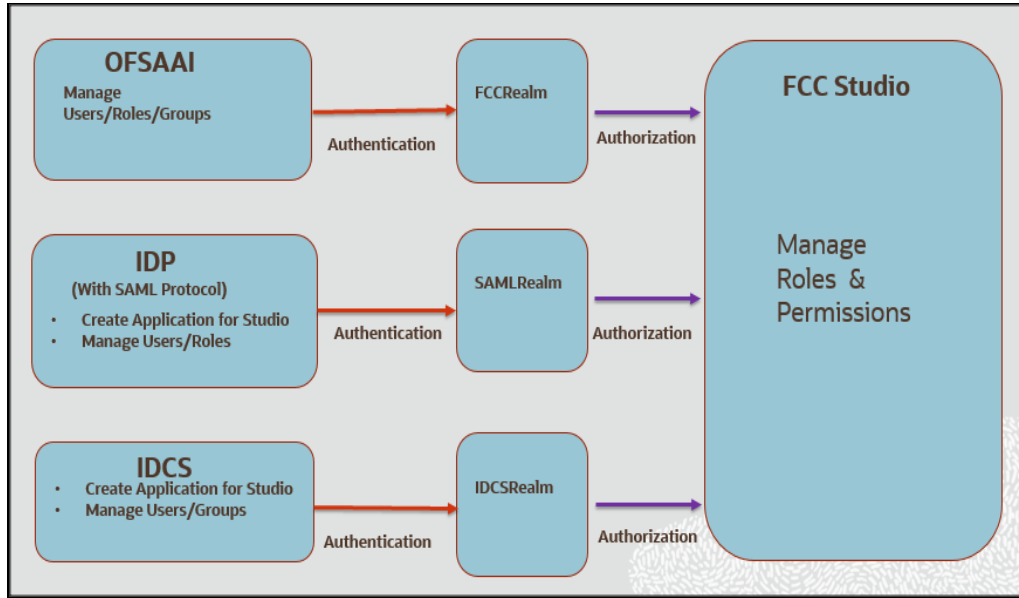
Realms (FCCRealm, SAMLRealm, and IDCSRealms) are selected based on the Identity Provider (IDP) during the installation. For more information, see [Oracle Financial Crime and Compliance Studio Application Installation Guide](#).

After you select the realms, you can register a set of schema objects or roles (secured objects) for realm protection and authorize a set of users or roles to access the secured objects.

The FCC Studio application is accessed using the following realms that you have selected during installation of FCC Studio application:

- **FCCMRealm:** The FCCMRealm uses Oracle Financial Services Analytical Applications Infrastructure (OFSAAI) Identity Management system for user authentication. Users, Roles, and Groups are created in the OFSAAI. The OFSAAI facilitates System Administrators to provide access, monitor, and administer users along with the Infrastructure metadata operations. The required permissions to roles or groups are authorized in the FCC Studio applications using the [Permission](#) feature.
- **SAMLRealm:** The SAMLRealm uses an identity provider (IDP) Identity Management system for user authentication. Security Assertion Markup Language (SAML) is an open standard that allows identity providers (IDP) to pass authorization credentials to service providers (SP). IDP acts as the Single Sign-On (SSO) service. Users and Roles are created in the IDP. The required permissions to Users and Roles are authorized in the FCC Studio applications using the [Permission](#) feature.
- **IDCRealm:** The IDCRealm uses Oracle Identity Cloud Service (IDCS) Identity Management system for user authentication. IDCS is Oracle's next-generation security and identity management platform that is cloud-native and designed to be an integral part of the enterprise security fabric, providing modern identity for modern applications. Users and Groups are created in the IDCS. The required permissions to Users and Groups are authorized in the FCC Studio applications using the [Permission](#) feature.

The following image illustrates the authentication and authorization process in FCC Studio.



3.1 Access FCC Studio Using FCCRealm

This section provides information on the creation of users who can access FCC Studio using the FCCMRealm method of authentication through Oracle Financial Services Analytical Applications Infrastructure (OFSAAI). The users with SYSADMN and SYSAUTH roles in OFSAAI can create and authorize users respectively.

Identity Management in the OFSAAI facilitates System Administrators to provide access, monitor, and administer users along with the Infrastructure metadata operations. The Security Management System (SMS) component is incorporated with Password Encryption, Single Logon, Role and Data-Based Security, Access Control, and Audit Trail feature to provide a highly flexible security envelope. Administrators can create, map, and authorize users defining a security framework that can restrict access to the data and meta-data in the warehouse, based on a fine-grained access control mechanism. These activities are done at the initial stage and then on a required basis. For more information on creating and authorizing users in the following applications:

- For OFS ECM, see **Managing User Administration and Security Configuration** section in the Oracle Financial Services Case Management Enterprise Administration and Configuration Guide.
- For BD, see **Managing User Administration and Security Configuration** section in the Oracle Financial Services Behavior Detection Administration Guide.

In addition, creating and authorizing users in OFSAAI, see the [Oracle Financial Services Analytical Applications Infrastructure User Guide](#).

The following table describes the ready-to-use roles and the corresponding user groups who can access FCC Studio using FCCMRealm.

NOTE Only in FCCRealm, users are mapped to user groups.

The default permissions mapped to these users and user groups are available in the [Permission](#) section. However, these permissions can be added or modified. [Table 1](#) describes Roles and User Groups in FCC Studio.

Role	User Groups
DSADMIN	DSADMINGRP
DSINTER	DSINTERGRP
DSUSER	DSUSERGRP
DSBATCH	DSBATCHGRP
DSAPPROVE	DSAPPROVEGRP
DSREDACT	DSREDACTGRP

3.2 Access FCC Studio Using SAMLRealm

This section provides information on managing users who can access FCC Studio with Identity Provider (IdP or IDP). The IdP acts as the Single Sign-On (SSO) service provider for implementations between FCC Studio, Investigation Hub, and Enterprise Case Management. This configuration prevents separate login for each application.

An identity provider (IdP) is a service that stores and verifies user identity. IdPs are cloud-hosted services, and they often work with single sign-on (SSO) providers to authenticate users. An identity provider (IdP or IDP) stores and manages users' digital identities. An IdP checks user identities via username-password combinations and other factors, or it may simply provide a list of user identities that another service provider (like an SSO) checks.

The following are the ready-to-use roles that can access FCC Studio using SAMLRealm.

To integrate FCC Studio with IdP as the SSO provider, follow these steps:

1. Create the following roles in the IDP system:
 - DSADMIN
 - DSINTER
 - DSUSER
 - DSBATCH
 - DSAPPROVE
 - DSREDACT
2. Map the user groups to the respective user based on the user roles.

The default permissions mapped to these users are available in the [Permission](#) section. However, these permissions can be added or modified.

3.3 Access FCC Studio Using IDCSRealm

This section provides information on managing users who can access FCC Studio with Oracle Identity Cloud Service (IDCS). The IDCS acts as the Single Sign-On (SSO) service provider for implementations between FCC Studio, Investigation Hub, and Enterprise Case Management. This configuration prevents separate login for each application. IDCS Realm Integration with FCC Studio allows SSO for both FCC Studio and ECM to provide seamless integration and eliminates the requirement to log in separately to FCC Studio.

The Oracle Cloud Infrastructure Console provides an integration with Oracle Identity Cloud Service (IDCS) that lets you perform many management tasks for your IDCS users and groups in the Console. In the Console, you can add users, remove users, add users to groups, assign roles to users to access services and instances, and reset the user password.

To manage Oracle Identity Cloud Service users and groups in the Console, you must have granted permissions in both the Oracle Cloud Infrastructure service and in Oracle Identity Cloud Service. To create users and groups in the Oracle Identity Cloud Service federation, you must have the Identity Domain Administrator role or be a member of a group that has been granted that role.

The following are the ready-to-use User Groups who can access FCC Studio using IDCSRealm.

To integrate FCC Studio with IDCS as the SSO provider, follow these steps:

1. Create the following user groups in the IDCS application:
 - DSADMIN
 - DSINTER
 - DSUSER
 - DSBATCH
 - DSREDACT
 - DSAPPROVE
2. Map the user groups to the respective user based on the user roles.

The default permissions mapped to these users are available in the [Permission](#) section. However, these permissions can be added or modified.

For more information on Managing Users with Oracle Identity Cloud Service Accounts, visit [Managing Users](#).

NOTE

It is recommended to use **FCCRealm** or **SAMLRealm**.

3.4 Access FCC Studio with MMG

This section provides information on managing users who can access FCC Studio with MMG. To access FCC Studio with MMG, the MMG user, should map to the following user roles based on the requirement:

- DSADMIN

To know more about MMG integration, see [FCC Studio Installation Guide](#). Also, to know how you can use MMG with Studio, see [FCC Studio User Guide](#). For more details, about MMG, see [MMG User Documentation](#).

3.5 Manage Permissions

The FCC Studio offers a rich permission system. The four key components are permissions, permission templates, roles, and groups. A permission is an action that applies to entities and types. A permission can be a general action, an action on all entities of a particular type, or an action on a specific entity of a type.

Authenticated users access FCC Studio through FCCRealm, SAMLRealm, and IDCSRealm. Users or User Groups are mapped to roles (DSADMIN, DSINTER, and so on) in the respective IDP (Identity Provider). Each role group is assigned to certain permissions (to use general, notebook, graph, interpreters, and so on). And, every permission has certain actions to perform (delete a paragraph, modify paragraph, execute interpreter variant, and so on). Hence, users or user groups mapped to roles can perform only assigned actions to which they have permissions.

- **User:** Authorized users can access FCC Studio through realms. For example, FCCMADMIN can access FCC Studio through FCCRealm or IDCSRealm.
- **Role:** Users are mapped to one or more roles by the realms. For example, FCCMADMIN is mapped to DSUSER role. A role consists of one or more permissions.
- **Permission:** Permission is a set of actions. For example, a permissions to perform on Notebook, Graph, Interpreter Variant, Interpreter, General, and so on. Having permissions on Notebooks, you can delete a notebook, schedule a notebook, and so on.
- **Action:** Actions are read, write, execute, and so on.

3.6 Work with Permissions

Use this section to view the logged in users and view, add, or modify ready-to-use permissions granted to the users, roles, or groups. You can also create groups, roles, and permission templates (actions).

NOTE You can only view users and their details.

It is recommended to:

- Use the Roles for permissions mapping.
- Use the Groups and Permission Templates that are configured and ready-to-use in the application.


To manage the permissions, log in to the FCC Studio application and follow these steps:

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

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

`<Port_Number>` for,

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

2. Enter the System Administrator Username and Password.
3. Click Login. The Crime and Compliance Studio application's landing page is displayed.
4. Click the Navigation Menu  on the upper-left corner. The applicable menu items to the logged-in user are displayed.
5. In the Crime and Compliance LHS Menu, select Permissions.

In the Permissions page, you have the following panes:

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

3.6.1 Users

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

Users						
All users that have logged in at least once.						
Username	Last Login	Roles	Groups	Permissions	Creation Time	Actions
FCCMDSUSER	23 Aug 2020 12:00 pm	DSUSER, DSREDACT	-		23 Aug 2020 12:00 pm	
FCCMDSADMIN	26 Aug 2020 05:42 pm	DSADMIN	-		19 Aug 2020 05:05 pm	

Page 1 of 1 (1-2 of 2 items) < 1 >

3.6.2 Groups

A Group consists of one or more permissions. A user can belong to multiple groups. For example, Oracle Labs, General, and External. The Groups section allows users to view and manage all group and permissions for it. Groups can be added, updated, and deleted. However, it is recommended to use ready-to-use groups in FCC Studio.

NOTE

Groups are applicable if you are using IDCSRealm.

Groups

Groups are sets of users. Groups can have permissions assigned to them just like permissions can be assigned to users. Users inherit the permissions from all the groups they are in. Group assignment is controlled by the application.

Name	Permissions	Creation Time	Actions
DSREDACT	create_notebook, import_notebook, create_notebook, import_notebook, create_notebook, import_notebook	19 Aug 2020 05:04 pm	
DSAPPROVER	interpreter_variant_view, interpreter_variant_execute, interpreter_variant_execute, interpreter_variant_view, interpreter_variant_execute, interpreter_variant_view, interpreter_variant_execute, interpreter_variant_execute, interpreter_variant_view	19 Aug 2020 05:04 pm	
DSINTER	interpreter_variant_view, interpreter_variant_execute, interpreter_variant_execute, interpreter_variant_view, interpreter_variant_execute, interpreter_variant_view, interpreter_variant_execute, interpreter_variant_view	19 Aug 2020 05:04 pm	

3.6.3 Roles

A Role consists of one or more permissions. A user can have multiple roles that are assigned in IDP. For example, admin, user, and guest. All the listed roles can be added, updated, and deleted. However, it is recommended to use ready-to-use roles in the application.

NOTE Roles are applicable if you are using FCCRealm and SAMLRealm.

Roles

Roles are sets of users. Roles can have permissions assigned to them just like permissions can be assigned to users. Users inherit the permissions from all the roles they have. Role assignment is controlled by the realm.

Name	Permissions	Creation Time	Actions
dfdsfdsfdsfdsfdsfdsf	create_permission_template	20 Aug 2020 01:59 pm	
DSREDACT	create_notebook, review_request, import_notebook, create_notebook, review_request, import_notebook, create_notebook, review_request, import_notebook, create_notebook	19 Aug 2020 05:04 pm	
DSAPPROVER	interpreter_variant_view, interpreter_variant_execute, interpreter_variant_execute, interpreter_variant_view, interpreter_variant_execute, interpreter_variant_view, interpreter_variant_execute, interpreter_variant_execute, interpreter_variant_view	19 Aug 2020 05:04 pm	

3.6.4 Permission Templates

A Permission Template is a set of actions. For example, limited_read, read, and write. All the listed Permission Templates can be added, updated, and deleted. However, it is not recommended to use a ready-to-use permission template in the application.

Permission Templates		+ Create Permission Template	
Set of (often logically related) actions to simplify applying multiple permissions (e.g. when assigning permissions in the "Share Notebook" dialog)			
Name	Permissions	Creation Time	Actions
own		20 Aug 2020 01:36 pm	
write	iframe paragraph_execute update paragraph_modify clone run_all view invalidate_session clear export Show More ...	20 Aug 2020 01:36 pm	
read	iframe view_code view snapshot view_result clone export template paragraph_view layout Show More ...	20 Aug 2020 01:36 pm	
limited_read	view_result layout paragraph_view view_code view export template iframe import_notebook create_notebook	20 Aug 2020 01:36 pm	

Manage ready-to-use Groups, Roles, and Permission Templates

To modify ready-to-use Groups, Roles, and Permission Templates, follow these steps:

1. In the Group pane, place your cursor in the Action column against required Groups, Roles, and Permission Templates name. The Modify and Delete icons are displayed.
2. To modify ready-to-use Groups, Roles, and Permission Templates, click Modify . The Update Groups, Update Roles, or Update Permission Templates window is displayed. Go to required options, such as Permission, Notebook, Interpreter Variant, General, Interpreter, Graph, and so on.
3. Click the required permissions such as create a permission template, delete a paragraph, and so on. The selected permissions are underlined.
4. To select all the options, click All . All the options in the Update Group window are selected. To deselect all the options, click All again.
5. To delete the permissions, select the required permission and click Delete . The selected permission is deleted.
6. Click Save. A confirmation message is displayed.

3.6.5 Create New Groups, Roles, and Permission Templates



When new users, roles, or groups are created in the respective realms, you must authorize them with permissions. Use this section to authorize new groups, roles, or permission templates.

To create Groups, Roles, and Permission Templates, follow these steps:

1. In the Groups, Roles, and Permission Templates pane. Click Create . The Create window for Groups, Roles, or Permission Templates is displayed.
2. Enter the name of Groups, Roles, or Permission Templates.

NOTE Enter up to 30 characters.

3. Select the required Permissions, Notebooks, Interpreter Variants, General actions, Interpreters, Graphs. The selected permissions are underlined.

4. To select all the options, click All . All the options in the Create window are selected. To deselect all the options, click All  again.
5. Click Save. A confirmation message is displayed.

3.7 List of Permissions

FCC Studio offers a set of permissions in the form of actions that can be applied to types.

The table below defines the types managed by FCC Studio:

Type	Description
all	affects all types
general	not directly related to a type
graph	affects graphs
job	affects jobs
interpreter	affects interpreters
interpreter_variant	affects interpreter variants
notebook	affects notebooks and paragraphs
permission	affects permission management view

The table below groups the available actions by type:

Type	Description	Allows user to
all	*	Do all of the below
general	create_notebook	Create a notebook
general	approve	Approving Scenario Notebooks
general	review_request	Permission for sending Manual Edges for Review
general	review_approve	Approve Manual Edges permission
general	delete_all	Delete all notebooks in the workspace view

general	export_all	Export all notebooks in the Workspace view
general	graph_create	Create a graph in the Graphs tab
general	import_notebook	Import a notebook
general	view_dashboard_tab	View the Tasks tab
general	view_permissions_tab	View the Permissions tab
general	view_interpreter_tab	View the Interpreters tab
general	view_credentials_tab	View the Credentials tab
general	create_credential	Create a credential
graph	graph_delete	Delete a graph
graph	graph_share	Share a graph
graph	graph_update	Update a graph
graph	graph_view	View a graph
interpreter	interpreter_create_variant	Create a new interpreter variant
interpreter	interpreter_update_variant	Update a variant of an interpreter
interpreter	interpreter_view	View an interpreter
interpreter_variant	interpreter_variant_execute	Execute an interpreter variant
interpreter_variant	interpreter_variant_delete	Delete an interpreter variant
interpreter_variant	interpreter_variant_view	View an interpreter variant
job	job_cancel	Cancel a job
job	job_view	View a job
notebook	add_relation	Add a relation to a notebook
notebook	attach	(Deprecated) Attach a notebook

notebook	clear	Clear all results in a notebook
notebook	clone	Clone a notebook
notebook	delete	Delete a notebook
notebook	detach	(Deprecated) Detach a notebook
notebook	export	Export a notebook
notebook	iframe	Open a notebook in iframe view
notebook	invalidate_session	Invalidate the session of a notebook
notebook	layout	Change the layout of a notebook
notebook	paragraph_comment	Comment on paragraphs in a notebook
notebook	paragraph_create	Create a new paragraph in a notebook
notebook	paragraph_delete	Delete the paragraphs in a notebook
notebook	paragraph_execute	Execute the paragraphs in a notebook
notebook	paragraph_modify	Modify the paragraphs in a notebook
notebook	paragraph_move	Move the paragraphs in a notebook
notebook	paragraph_view	View the paragraphs in a notebook
notebook	remove_relation	Remove a relation from a notebook
notebook	rename	Rename a notebook
notebook	run_all	Run all paragraphs in a notebook
notebook	schedule_notebook	Schedule a notebook
notebook	share	Share a notebook

notebook	set_readonly	Set a notebook to read-only
notebook	snapshot	Take a snapshot (immutable copy) of a notebook
notebook	style	Change the style of a notebook
notebook	template	Add a template to a notebook
notebook	toggle_show_code	Toggle the Show Code button in a notebook
notebook	toggle_show_result	Toggle the Show Result button in a notebook
notebook	update	Update a notebook
notebook	view	View a notebook
notebook	view_code	View the code of the paragraphs of a notebook
notebook	view_result	View the result of the paragraphs of a notebook
notebook	view_sessions	View the sessions of a notebook
permission	create_group	Create a group
permission	create_permission_template	Create a permission template
permission	create_role	Create a role
permission	delete_group	Delete a group
permission	delete_permission_template	Delete a permission template
permission	delete_role	Delete a role
permission	update_group	Update a group
permission	update_permission_template	Update a permission template
permission	update_role	Update a role

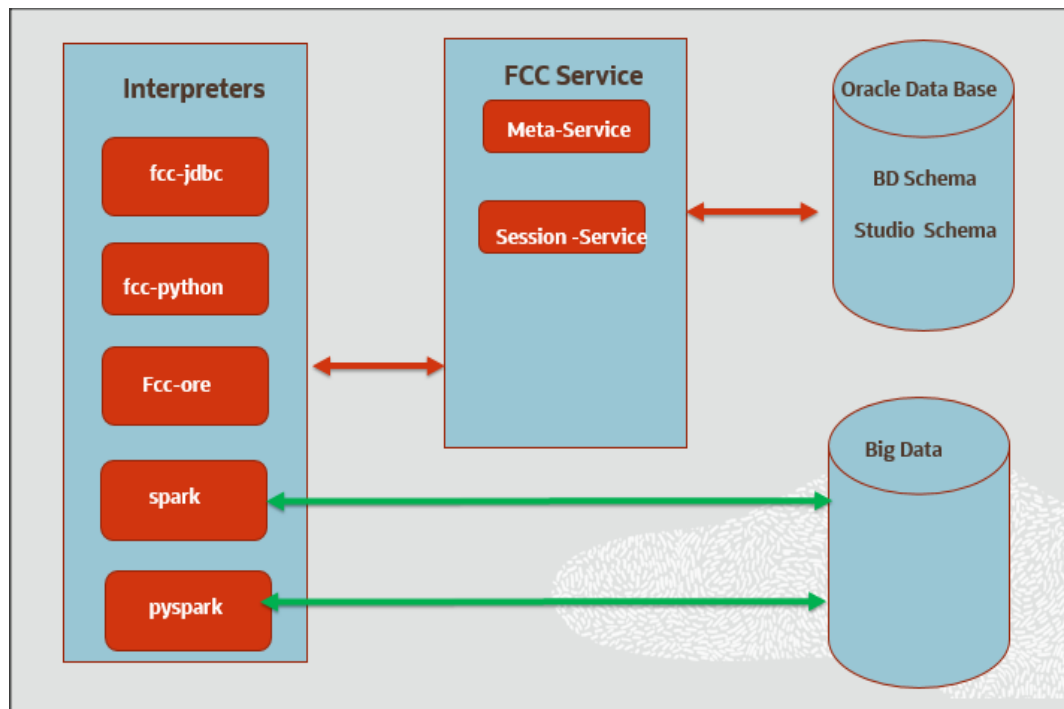
permission	update_user	Update a user
permission	view_group	View the Groups section in the Permissions screen
permission	view_permission_template	View the Permission Templates section in the Permissions screen
permission	view_role	View the Roles section in the Permissions screen
permission	view_user	View the Users section in the Permissions screen
credential	view_credential	View a credential and download its file in the credentials screen
credential	use_credential	Use a credential to connect to a datasource
credential	delete_credential	Delete a credential from the credentials screen

4 Interpreter Configuration and Connectivity

An interpreter is a program that directly executes instructions written in a programming or scripting language, without requiring them previously to be compiled into a machine language program. Interpreters are plug-ins, which enable users to use a specific language to process data in the back-end. Examples of Interpreters are, jdbc-interpreter, spark-interpreters, python-interpreters, and so on. Interpreters allow you to define customized drivers, URLs, passwords, connections, SQL result to display, and so on.

In FCC Studio, Interpreters are used in Notebooks to execute code in different languages. Each interpreter has a set of properties that are adjusted and applied across all notebooks. For example, by using the python-interpreter, it is possible to change between versions, whereas the jdbc-interpreter offers to customize the URL, schema, or credentials. In FCC Studio, you can either use a default interpreter variant or create a new variant for an interpreter. You can create more than one variant for an interpreter. The benefit of creating multiple variants for an Interpreter is to connect different versions of interpreters (Python ver: 3, Python ver: 2, and so on), this helps to connect a different set of users, database schema. For example, FCC Studio schema, BD schema, and so on. FCC Studio provides secure and safe credential management such as Oracle Wallet (jdbc wallet) Password (jdbc password), or KeyStores to link to interpreter variants to access secured data.

The following image illustrates the examples of interpreters used in FCC Studio and database connections.



Topics:

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

- [Link Credentials](#)
- [Create Credential](#)
- [Create an Interpreter Variant](#)
- [Modify the Python Docker Images](#)
- [Configure Spark Query Parameters](#)

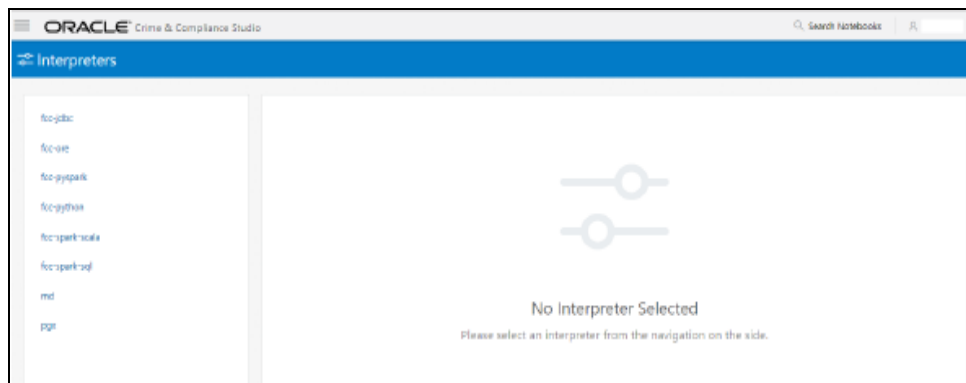
4.1 Configure Interpreters

FCC Studio has ready-to-use interpreters such as fcc-jdbc, fcc-spark-scala Interpreter, fcc-python Interpreter, and so on. You can configure them based on the use case. Additional variants of interpreters are created as multiple users might require different settings to access the database securely. Interpreters such as fcc-jdbc and jdbc are linked using credentials to enable secure data access.

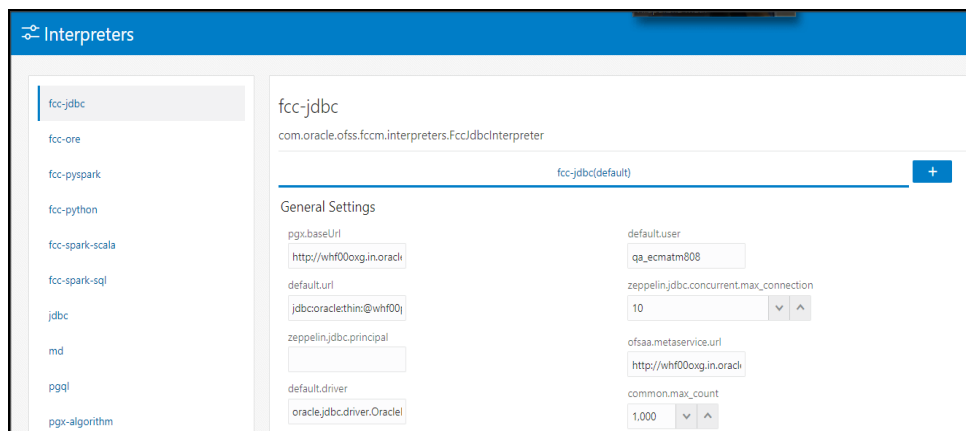
Interpreters are configured when you want to modify URL, data location, drivers, enable or disable connections, and so on.

To configure ready-to-use interpreters, follow these steps:

1. In the Crime & Compliance Studio menu list, click Interpreters. By default, the Interpreters page lists all the available interpreters.



2. Click the interpreter that you want to view from the list displayed on the LHS. The default configured interpreter variant is displayed on the RHS.



3. Modify the values in the fields as per requirement. For example, to modify the limit of a parameter, to connect to a different schema, PGX server, and so on.
4. Click Update. The modified values are updated in the interpreter.

[Table 2](#) lists the Ready-to-use interpreter in FCC Studio:

Interpreters	Description
fcc-jdbc Interpreter	<p>The fcc-jdbc Interpreter is a ready-to-use interpreter used to connect to Behavior Detection (BD) or Enterprise Case Management (ECM) atomic schema and is used for scenario notebooks.</p> <p>The parameters are configured to connect to an ECM or BD schema.</p> <p>NOTE: If it is used to connect to another schema, Virtual Private Database (VPD) must be configured. Alternatively, you can use jdbc interpreter.</p> <p>In the fcc-jdbc Interpreter, you can configure the connection to pull or push data to the desired location, set the default jdbc authentication type, link credentials, and so on. To access addition access permissions for fcc-jdbc Interpreter, you can link with credentials too.</p>
fcc-ore Interpreter	<p>The fcc-ore Interpreter is a ready-to-use interpreter used to connect to BD schema. This interpreter is used to write a notebook in R language (ORE: Oracle R Enterprise). Additional configuration is required and must be done as pre-requisite or post-installation with a manual update in the interpreter setting.</p> <p>In the fcc-ore Interpreter, you can configure the Oracle system identifier (SID) of the database server to configure different databases, configure to display the output number of rows, set the hostname of the database server to connect the fcc-ore Interpreter, schema details, and so on.</p>
fcc-pyspark Interpreter	<p>The fcc-pyspark Interpreter is a ready-to-use interpreter used to connect to Big data server from the livy server. After it is connected, you can write in the PySpark language to query and perform analytics on data present in big data.</p> <p>In the fcc-pyspark Interpreter, you can configure the number of executors to launch for the current session, cached execution timeout, amount of memory to use for the executor process, and so on.</p>
fcc-python Interpreter	<p>The fcc-python Interpreter is used to write Python code in a notebook to analyze data from different sources, machine learning, and artificial intelligence, and so on.</p> <p>In the fcc-python Interpreter, you can configure the python installed path, set the maximum number of results that must be displayed, change the Python version, add Python Packages, and so on.</p>
fcc-spark-scala Interpreter	<p>The fcc-spark-scala Interpreter is a ready-to-use interpreter using the livy server to connect to the Big data server. It is used to perform analytics on data present in Big data using the Scala language.</p> <p>In the fcc-spark-scala Interpreter, you can configure the number of executors to launch for the current session, set livy URL, and configure keytab location, and so on.</p>

<p>fcc-spark-sql Interpreter</p>	<p>The fcc-spark-sql Interpreter is a ready-to-use interpreter using the livy server to connect to the Big data cluster. It is used to perform analytics on data present in Hive schema using SQL queries.</p> <p>In the fcc-spark-sql Interpreter, you can configure the number of executors to launch for the current session, data pull interval in milliseconds, and so on.</p>
<p>jdbc Interpreter</p>	<p>The jdbc Interpreter is a ready-to-use interpreter used to connect to Studio schema. This interpreter is used to connect and write SQL queries on any schema without any restriction.</p> <p>In the jdbc Interpreter, you can configure schema details, link Wallet Credentials to jdbc Interpreter, and so on.</p>
<p>md Interpreter</p>	<p>The md Interpreter is used to configure the markdown parser type. This interpreter is used to display text based on Markdown, which is a lightweight markup language.</p> <p>The connection is not applicable for this interpreter.</p>
<p>pgql Interpreter</p>	<p>The pgql Interpreter is a ready-to-use interpreter used to connect to the configured PGX server. This interpreter is used to perform PGQL queries on the graph in FCC Studio.</p> <p>In the pgql Interpreter, you can configure the class which implements the formatting of the visualization output, the size of the output message, and so on.</p>
<p>pgx-algorithm Interpreter</p>	<p>The pgx-algorithm Interpreter is a ready-to-use interpreter used to connect to the configured PGX server. This interpreter is used to write the algorithm on the graph, and it is also used in pgx-java interpreter.</p> <p>In the pgx-algorithm Interpreter, you can configure the class which implements the PGQL driver, the size of the output message, and so on.</p>
<p>pgx-java Interpreter</p>	<p>The pgx-java Interpreter is a ready-to-use interpreter used to connect to the configured PGX server. This interpreter is used to write the algorithm on the graph and also used in the pgx-java interpreter.</p> <p>Configure the class which implements the formatting of the visualization output, the class which implements the PGQL driver, and so on.</p>
<p>pyspark Interpreter</p>	<p>The pyspark Interpreter connects to the big data environment by default. Users must write code for connection either in the Initialization section or in the notebook's paragraph.</p> <p>This interpreter is used to write the pyspark language to query and perform analytics on data present in big data. This requires additional configuration, which must be performed as prerequisite or as post-installation with the manual change of interpreter settings.</p> <p>In the pyspark Interpreter, you can configure the Python binary executable to use for PySpark in both driver and workers, set true to use IPython, else set to false, and so on.</p>

spark Interpreter	<p>The spark Interpreter connects to the big data environment by default. Users must write for connection either in the Initialization section or in the notebook’s paragraph.</p> <p>This interpreter is used to perform analytics on data present in the big data clusters in the Scala language. This requires additional configuration, which must be performed as a prerequisite or as post-installation with the manual change of interpreter settings.</p> <p>In the spark interpreter, you can configure the cluster manager to connect, print the Read–eval–print loop (REPL) output, the total number of cores to use, and so on.</p>
-----------------------------------	---

4.1.1 fcc-jdbc Interpreter

The fcc- jdbc is a ready-to-use Interpreter variant in FCC Studio that connects BD and ECM database schema and is used for scenario notebooks. The parameters are configured to connect different schemas. It filters results based on security attributes mapped to the user.

NOTE	If it is used to connect to another schema, Virtual Private Database (VPD) must be configured. Alternatively, you can use the jdbc Interpreter.
-------------	---

In the fcc-jdbc Interpreter, you can configure the connection to pull or push data to the desired location, set the default jdbc authentication type, link credentials, and so on. To access additional access permissions for the fcc-jdbc Interpreter, you can link with credentials too.

Use this section to perform the following activities:

- [Configure a fcc-jdbc Interpreter Variant](#)
- [Link Wallet Credentials to fcc-jdbc Interpreter](#)

4.1.1.1 Configure a fcc-jdbc Interpreter Variant

To configure an fcc-jdbc Interpreter variant, follow these steps:

1. On the Interpreter page LHS menu, select fcc-jdbc. The fcc-jdbc interpreter pane is displayed.
2. Enter the following information in the fcc-jdbc Interpreter variant pane as tabulated in [Table 3](#).

Field	Description
pgx.baseUrl	Enter the PGX URL. This is the location where the data is pushed. For example: <code>http://<HOSTNAME>:7007</code>

default.url	<p>Enter the ofsaa jdbc URL.</p> <p>For example:</p> <pre>jdbc:oracle:thin:@<database hostname>:<database port>:<SID></pre> <p>and</p> <pre>jdbc:oracle:thin:@<database hostname>:<database port>/<Service Name></pre> <p>NOTE:</p> <p>If you want to use the Oracle Wallet credentials, you must enter the alias name in the following format:</p> <pre>jdbc:oracle:thin:@<alias_name></pre>
zeppelin.jdbc.principal	<p>Enter the principal name to load from the keytab. This variable is used if they are connecting to HIVE from the jdbc Interpreter.</p>
default.driver	<p>Enter the default JDBC driver name.</p> <p>For example:</p> <pre>oracle.jdbc.driver.OracleDriver</pre>
default.completer.ttlInSeconds	<p>Enter the time to live SQL completer in seconds (-1 to update every time, 0 to disable update)</p>
default.password	<p>Enter the default password.</p> <p>NOTE:</p> <p>This value can be null if you have entered the alias name in the <code>default.url</code> parameter for the fcc-jdbc interpreter.</p>
default.splitQueries	<p>Enter 'True' or False' to specify the presence of default split queries. Each query is executed apart and returns the result.</p>
default.completer.schemaFilters	<p>Enter a comma-separated schema filter to get metadata for completions.</p>
ofsaa.sessionservice.url	<p>Enter the session service URL in this field.</p> <p>For example:</p> <pre>http://<HOSTNAME>:7047/sessionservice</pre> <p><HOSTNAME> refers to the server name or IP address where fcc-studio is installed.</p>
default.user	<p>Enter the name of the default user in this field.</p> <p>For example, root.</p>
zeppelin.jdbc.concurrent.max_connection	<p>Enter the number of maximum connections allowed.</p> <p>NOTE: This depends on the database settings.</p>

ofsa.metaservice.url	Enter the metaservice URL in this field. For example, <code>http://<HOSTNAME>:7045/metaservice</code> <HOSTNAME> refers to the server name or IP address where fcc-studio is installed.
common.max_count	Enter the maximum number of SQL result to display.
zeppelin.jdbc.auth.type	Enter the default JDBC authentication type.
zeppelin.jdbc.precode	Enter the snippet of code that executes after the initialization of the interpreter.
zeppelin.jdbc.concurrent.use	Enter to enable or disable concurrent use of JDBC connections. Enter 'True' or 'False'.
zeppelin.jdbc.keytab.location	Enter the keytab file location.

4.1.1.2 Link Wallet Credentials to fcc-jdbc Interpreter

FCC Studio provides secure and safe credential management. Examples for credentials are passwords, Oracle Wallets, or KeyStores.

Oracle Wallet is a file that stores database authentication and signing credentials. It allows users to securely access databases without providing credentials to third-party software, and easily connect to Oracle products.

A Keytab is a file containing pairs of Kerberos principals and encrypted keys (which are derived from the Kerberos password). You can use a keytab file to authenticate various remote systems using Kerberos without entering a password. However, when you change your Kerberos password, you must recreate all your keytab files.

Use this section to link credentials (a wallet or a password) to the fcc-jdbc interpreter variant to enable secure data access. This linking enables the fcc-jdbc interpreter to securely connect to the specified Oracle database. For more information to link Wallet Credentials to the fcc-jdbc Interpreter, see [Linking Credentials to Interpreters](#).

NOTE The Credentials' section is enabled if an interpreter variant can accept credentials.

You can also create new credentials and link to the fcc-jdbc Interpreter. For more information, see [Creating Credentials](#).

4.1.2 fcc-ore interpreter

The fcc- ore (ORE: Oracle R Enterprise) is a ready-to-use interpreter to connect to BD schema. This interpreter is used to write a notebook in R language, to perform R based analytics on data from database schema on the business table.

Additional configuration is required and must be done as a prerequisite or post-installation with a manual update in the interpreter setting. In the fcc-ore interpreter, you

can configure the Oracle system identifier (SID) of the database server to connect the fcc-ore Interpreter, configure to the display the output number of rows, set the hostname of the database server to connect the fcc-ore interpreter, schema details, and so on.

To configure the fcc-ore interpreter variant, follow these steps:

1. On the Interpreter page LHS menu, select fcc-ore. The fcc-ore interpreter pane is displayed.
2. Enter the following information in the fcc-ore interpreter variant pane as tabulated in [Table 4](#).

Field	Description
ore.sid	Enter the Oracle system identifier (SID) of the database server where the fcc-ore interpreter must be connected. A SID is a unique name that uniquely identifies the database instance whereas a service name is the Database TNS Alias that is given when users remotely connect to the database.
rendering.row.limit	Enter the number of rows to be shown in the fcc-ore interpreter output. For example, 1000.
ore.conn_string	Enter the database connection URL with which the fcc-ore interpreter can make the connection to the schema. NOTE: This is not a mandatory field.
https_proxy	Enter the proxy server to establish a connection with the internet. For example, <code>http://sample.proxy.com</code>
ore.type	Enter the fcc-ore interpreter type as Oracle.
ore.password	Enter the schema password where the fcc-ore interpreter is connected.
libpath	Enter the custom library path from where R packages are installed via FCC Studio and added to R lib (library) path. R packages are collections of functions and data sets. Enter the library path of the home directory where FCC Studio is installed. For example: If you want the packages to be available under <code>/home/user/library</code> , and FCC Studio is installed at <code>/home/user/fccstudio</code> , then mention <code>/library</code> as the lib path.
ore.host	Enter the hostname of the database server to connect with the fcc-ore interpreter.

rserve.password	<p>Enter the Rserve password.</p> <p>NOTE: Enter up to 255 characters.</p> <p>Rserve is an R package that allows other applications to talk to R using TCP/IP. It creates a socket server to which other applications can connect.</p>
rendering.numeric.format	<p>Enter the number of digits to round off.</p> <p>For example, <code>%.2f</code>.</p> <p>This displays the output. You can round off the digits in number after the decimal.</p>
ore.service_name	<p>Enter the service name of the database server to connect the fcc-ore interpreter.</p>
rserve.try.wrap	<p>Enter False.</p>
rserve.host	<p>Enter the Rserve host. The Rserve is a TCP/IP server which allows other programs to use facilities of R from various languages without initializing R or linking to the R library.</p>
repo_cran	<p>Enter the CRAN URL from where R libraries are downloaded to install R packages.</p> <p>For example, <code>https://cran.r-project.org/</code></p> <p>The Comprehensive R Archive Network CRAN is a network of web servers around the world where you can find the R source code.</p>
ofsaa.sessionservice.url	<p>Enter the session service URL.</p> <p>For example: <code>http://<HOSTNAME>:7047/sessionservice</code></p> <p>Here, <code><HOSTNAM></code> refers to the server name or IP address where fcc-studio is installed.</p>
ore.all	<p>Enter 'All' to sync all tables with the fcc-ore interpreter.</p> <p>The value must be 'True'.</p>
rserve.plain.qap.disabled	<p>Enter whether plain QAP is disabled on the server or not. If disabled, the connection will be always attempted using SSL.</p> <p>For example: 'False'.</p>
ore.user	<p>Enter the schema name where the fcc-ore interpreter is to be connected.</p>

http_proxy	Enter the proxy server to establish a connection with the internet. This value is used to set the initial setting that makes the environment compatible to download the libraries available in R. For example: <code>http://sample.proxy.com</code>
rserve.port	Enter the Rserve port.
rserve.secure.login	Enter 'True' to enforce a secure login.
rendering.knitr.options	Enter the Knitr output rendering option. For example: <code>out.format = 'html', comment = NA, echo = FALSE, results = 'verbatim', message = F, warning = F, dpi = 300</code> knitr is an engine for dynamic report generation with R. It is a package in the R programming language that enables integration of R code into LaTeX, LyX, HTML, and Markdown, AsciiDoc, and reStructuredText documents.
rserve.user	Enter the Rserve username.
ore.port	Enter the port number of the database server to connect with the fcc-ore interpreter.
ofsa.metaservice.url	Enter the metaservice URL. For example: <code>http://<HOSTNAME>:7045/metaservice</code> Here, <HOSTNAME> refers to the server name or IP address where fcc-studio is installed.
rendering.include.row.name	Specify whether to include row names. Enter 'True' to include or 'False' to exclude.
rendering.knitr.image.width	Enter the image width specification for ore output. For example, 60.

4.1.3 fcc-pyspark Interpreter

The fcc-pyspark Interpreter is a ready-to-use interpreter used to connect to Big data server from the livy server. After it is connected, you can write in the pyspark language to query and perform analytics on data present in the Big data. In the fcc-pyspark interpreter, you can configure the number of executors to launch for the current session, cached execution timeout, amount of memory to use for the executor process, and so on.

To configure the fcc-pysprk interpreter variant, follow these steps:

1. On the Interpreter page LHS menu, select fcc-pyspark. The fcc-pyspark interpreter pane is displayed.

2. Enter the following information in the fcc-pyspark interpreter variant pane as tabulated in [Table 5](#).

Field	Interpreter
pgx.baseUrl	Enter the PGX Base URL. This is the location where the data is pushed. For example, <code>http://##HOSTNAME##:7007</code>
livy.spark.executor.instances	Enter the number of executors to launch for the current session. For example, Executor instances can be 1, 4, and so on.
livy.spark.dynamicAllocation.cachedExecutorIdleTimeout	Enter the cached execution timeout in seconds. Remove an executor that has cached data blocks.
zeppelin.livy.url	Enter the Livy URL. The Livy is an interface between Data Studio and Spark. This is the URL where the livy server is running. For example, <code>http://<hostname>:<port></code>
zeppelin.livy.pull_status.interval.millis	Enter the data pull interval in milliseconds. This is the interval for checking paragraph execution status.
livy.spark.executor.memory	Enter the amount of memory to use for the executor process. Executor memory per worker instance. For example, 512m, 32g.
livy.spark.dynamicAllocation.enabled	Specify whether the dynamic allocation is enabled or not. Enter 'True' to enable or 'False' to disable.
livy.spark.dynamicAllocation.minExecutors	Enter the minimum number of required dynamic allocation executors. It is the lower bound for the number of executors.
livy.spark.executor.cores	Enter the number of executor cores to use for the driver process. This is the number of cores per executor. For example, 1, 4, and so on.
zeppelin.livy.session.create_timeout	Enter the Zeppelin session creation timeout in seconds. This is the timeout in seconds for session creation.
zeppelin.livy.spark.sql.maxResult	Enter the maximum number of results that must be displayed.

livy.spark.jars.packages	Enter to add extra libraries to a livy interpreter.
livy.spark.driver.cores	Enter the number of driver cores to use for the driver process.
zeppelin.livy.displayAppInfo	Specify whether the application information must be displayed or not. Enter "True" or "False".
livy.spark.driver.memory	Enter the amount of memory to use for the driver process.
zeppelin.livy.principal	Enter the principal name to load from the keytab file.
ofsa.sessionservice.url	Enter the session service URL in this field. For example, <code>http://##HOSTNAME##:7047/session service</code> Here, ##HOSTNAME## refers to the server name or IP address where fcc-studio is installed.
ofsa.metaservice.url	Enter the metaservice URL in this field. For example, <code>http://##HOSTNAME##:7045/metaser vice</code> Here, ##HOSTNAME## refers to the server name or IP address where fcc-studio is installed.
zeppelin.livy.keytab	Enter the keytab file location.
livy.spark.dynamicAllocation.maxExecutors	Enter the maximum number of required dynamic allocation executors.

4.1.4 fcc-python Interpreter

The fcc-python interpreter is used to write Python code in a notebook to analyze data from different sources, machine learning, and artificial intelligence, and so on. In the fcc-python interpreter, you can configure the python installed path, set the maximum number of results that must be displayed, change the Python version, add Python Packages, and so on. When FCC Studio stops supporting any of the outdated Python versions, you can add or change the Python version. For example, Python 3.6.0, Python 3.6.6, Python 3.6.7, and so on.

Topics:

- [Configure an fcc-python Interpreter](#)
- [Change Python Version in the fcc-python Interpreter](#)

- [Add or Modify Python Packages to the fcc-python Interpreter](#)

4.1.4.1 Configure an fcc-python Interpreter

To configure an fcc-python interpreter variant, follow these steps:

1. On the Interpreter page LHS menu, select fcc-python. The fcc-python interpreter pane is displayed.
2. Enter the following information in the fcc-python interpreter variant pane as tabulated in [Table 6](#).

Field	Description
zeppelin.python	Enter the Python installed path. The value points to the default Python version set for the interpreter. NOTE: To use a different Python version, see Changing Python Version in the fcc-python Interpreter
zeppelin.python.useIPython	Set to 'True' to use IPython, else set to 'False'.
zeppelin.python.maxResult	Enter the maximum number of results that must be displayed.

4.1.4.2 Change Python Version in the fcc-python Interpreter

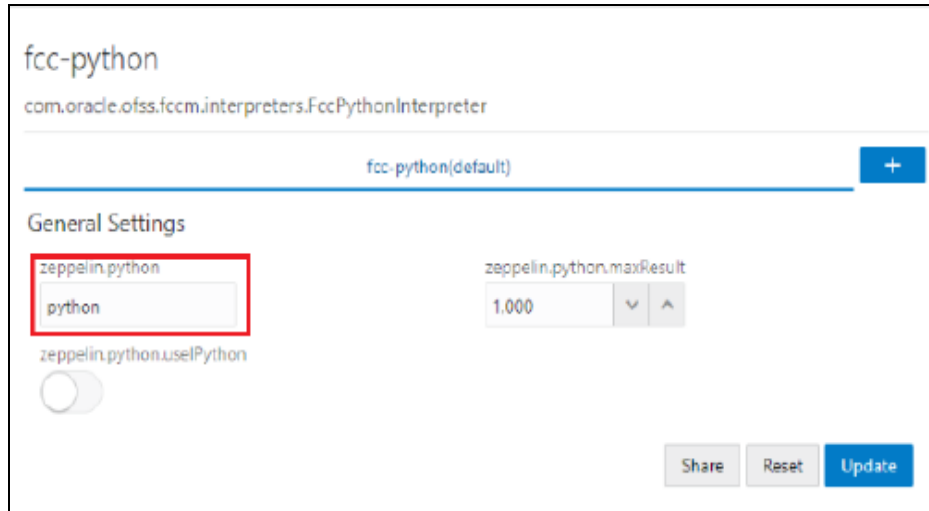
In the fcc-python interpreter, the Linux console uses the default python version in `/user/fccstudio/python_user/bin/python` as value. If you want to modify the python version, either you can create an interpreter variant or modify the existing python version in the same interpreter variant.

NOTE

Python2 is the default version used in the Linux console and it is no more supported. Hence, you can use any version of python3 or any virtual environment with a specific python version or a specific version of python packages.

To use a different version of Python, follow these steps:

1. Navigate to the fcc-python Interpreter Settings page.
2. Change the default Python version in the `zeppelin.python` parameter to the new version. For example, python3.6.



3. Create a new interpreter variant and configure the version in the `zeppelin.python` parameter. For information on creating a new interpreter variant, see [Creating a New Interpreter Variant](#). For example, to use Python 3.6, create a new `fcc-python` interpreter variant and enter the value as `python3.6`.

4.1.4.3 Add or Modify Python Packages to the fcc-python Interpreter

When a user wants to write something in Python but the packages are not present. Use case: ML or AI code. By default, the Linux server (or docker image) has a limited number of packages present inside it.

To add desired Python packages to the `fcc-python` interpreter, follow these steps:

- For FCC Studio installed on-premise:

To add or modify Python libraries to the `fcc-python` interpreter, contact System Administrator to install the required additional Python libraries on the Processing Server (Studio Notebook Server). The newly added Python libraries must be accessible to the Linux user for FCC Studio.

To add the python packages for `python3`, follow these steps:

1. Navigate to the `<Studio_Installation_Path>/python-packages/bin` directory.
2. Execute the following command:

```
python3 -m pip install <package name> --user
```

- For FCC Studio installed using Kubernetes:

To install additional Python libraries to the `fcc-python` interpreter, see [Modify the Python Images for the Python Interpreter](#).

4.1.5 fcc-spark-scala Interpreter

`fcc-spark-scala` Interpreter is a ready-to-use interpreter using the livy server to connect to the Big data server. It is used to perform analytics on data present in Big data using the Scala language. `fcc-spark-scala` interpreter does not connect to any schema by default.

Users must write code for connection either in the Initialization section or in the notebook's paragraph. In the fcc-spark-scala interpreter, you can configure the number of executors to launch for the current session, set the Livy URL, and configure keytab location, and so on.

To configure the fcc-spark-scala interpreter variant, follow these steps:

1. On the Interpreter page LHS menu, select fcc-spark-scala. The fcc-spark-scala interpreter pane is displayed.
2. Enter the following information in the fcc-spark-scala interpreter variant pane as tabulated in [Table 7](#).

Field	Description
pgx.baseUrl	Enter the PGX Base URL. This is the location where the data is pushed. For example, <code>http://<HOSTNAME>:7007</code>
livy.spark.executor.instances	Enter the number of executors to launch for the current session. For example, executor instances 1, 4, and so on.
livy.spark.dynamicAllocation.cachedExecutorIdleTimeout	Enter the cached execution timeout in seconds.
zeppelin.livy.url	Enter the Livy URL in this field. Livy is an interface between Data Studio and Spark. For example: <code>http://<hostname>:<port></code>
zeppelin.livy.pull_status.interval.millis	Enter the data pull interval in milliseconds. This is the interval for checking paragraph execution status.
livy.spark.executor.memory	Enter the amount of memory to use for the executor process. Executor memory per worker instance. For example, 512m and 32g.
livy.spark.dynamicAllocation.enabled	Specify whether the dynamic allocation is enabled or not. Enter 'True' to enable or 'False' to disable.
livy.spark.dynamicAllocation.minExecutors	Enter the minimum number of required dynamic allocation executors. It is the lower bound for the number of executors.
livy.spark.executor.cores	Enter the number of executor cores to use for the driver process. This is the number of cores per executor. For example, 1, 4, and so on.

zeppelin.livy.session.create_timeout	Enter the Zeppelin session creation the timeout in seconds. This is the timeout in seconds for session creation.
zeppelin.livy.spark.sql.maxResult	Enter the maximum number of results that must be displayed.
livy.spark.jars.packages	Enter to add extra libraries to the livy interpreter.
livy.spark.driver.cores	Enter the number of driver cores to use for the driver process.
zeppelin.livy.displayAppInfo	Specify whether the application information must be displayed or not. Enter 'True' to display or 'False' not to display.
livy.spark.driver.memory	Enter the amount of memory to use for the driver process.
zeppelin.livy.principal	Enter the principal name to load from the keytab file.
ofsa.sessionservice.url	Enter the session service URL in this field. For example, <code>http://<HOSTNAME>:7047/sessionservice</code> Here, <HOSTNAME> refers to the server name or IP where fcc-studio is installed.
ofsa.metaservice.url	Enter the metaservice URL in this field. For example, <code>http://<HOSTNAME>:7045/metaservice</code> Here, <HOSTNAME> refers to the server name or IP address where fcc-studio is installed.
zeppelin.livy.keytab	Enter the keytab file location.
livy.spark.dynamicAllocation.maxExecutors	Enter the maximum number of required dynamic allocation executors.
livy.spark.dynamicAllocation.initialExecutors	Enter the initial dynamic allocation executors.

4.1.6 fcc-spark-sql Interpreter

The fcc-spark-sql Interpreter is a ready-to-use interpreter uses livy server to connect Big data cluster. It is used to perform analytics on data present in Hive schema using Hive

queries. In the fcc-spark-sql interpreter, you can configure the number of executors to launch for the current session, data pull interval in milliseconds, and so on.

To configure the fcc-sprk-sql interpreter variant, follow these steps:

1. On the Interpreter page LHS menu, select fcc-sprk-sql. The fcc-sprk-sql interpreter pane is displayed.
2. Enter the following information in the fcc-sprk-sql interpreter variant pane as tabulated in [Table 8](#).

Field	Description
pgx.baseUrl	Enter the PGX Base URL. This is the location where the data is pushed. For example: <code>http://<HOSTNAME>:7007</code>
livy.spark.executor.instances	Enter the number of executors to launch for the current session. For example, Executor Instances can be 1, 4, and so on.
livy.spark.dynamicAllocation.cachedExecutorIdleTimeout	Enter the cached execution timeout in seconds.
zeppelin.livy.url	Enter the Livy URL. The Livy is an interface between Data Studio and Spark. For example: <code>http://<HOSTNAME>:8998</code>
zeppelin.livy.pull_status.interval.millis	Enter the data pull interval in milliseconds. This is the interval for checking paragraph execution status.
livy.spark.executor.memory	Enter the amount of memory to use for the executor process. Executor memory per worker instance. For example, 512m and 32g.
livy.spark.dynamicAllocation.enabled	Specify whether the dynamic allocation is enabled or not. Enter 'True' to enable or 'False' to disable.
livy.spark.dynamicAllocation.minExecutors	Enter the minimum number of required dynamic allocation executors. It is the lower bound for the number of executors.
livy.spark.executor.cores	Enter the number of executor cores to use for the driver process.
zeppelin.livy.session.create_timeout	Enter the Zeppelin session creation timeout in seconds.

zeppelin.livy.spark.sql.maxResult	Enter the maximum number of results that must be fetched.
zeppelin.livy.spark.sql.field.truncate	Specify to truncate values longer than 20 characters. Enter 'True' or 'False'.
livy.spark.jars.packages	Enter to add extra libraries to a livy interpreter.
livy.spark.driver.cores	Enter the number of driver cores to use for the driver process.
zeppelin.livy.displayApplInfo	Specify whether the application information must be displayed. Enter 'True' to display or 'False' to not to display.
livy.spark.driver.memory	Enter the amount of memory to use for the driver process.
zeppelin.livy.principal	Enter the principal name to lead from the keytab file.
ofsaa.sessionservice.url	Enter the session service URL in this field. For example: <code>http://<HOSTNAME>:7047/sessionservice</code> Here, <HOSTNAME> refers to the server name or IP address where fcc-studio will be installed.
ofsaa.metaservice.url	Enter the metaservice URL in this field. For example: <code>http://<HOSTNAME>:7045/metaservice</code> Here, <HOSTNAME> refers to the server name or IP address where fcc-studio will be installed.
zeppelin.livy.keytab	Enter the keytab file location.
livy.spark.dynamicAllocation.maxExecutors	Enter the maximum number of required dynamic allocation executors.

4.1.7 jdbc Interpreter

The jdbc Interpreter is a ready-to-use interpreter used to connect Studio schema without OFSAA. This interpreter is used to connect and write SQL queries on any schema without any restriction. The jdbc interpreter has no security attributes, it can be used to access any schema. In jdbc interpreter, you can configure schema details, link Wallet Credentials to the jdbc Interpreter, and so on.

Topics:

- [Configure jdbc Interpreter Variant](#)
- [Link Wallet Credentials to jdbc Interpreter](#)

4.1.7.1 Configure a jdbc Interpreter Variant

To configure a jdbc interpreter variant, follow these steps:

1. On the Interpreter page LHS menu, select jdbc. The jdbc interpreter pane is displayed.
2. Enter the following information in the jdbc interpreter variant pane as tabulated in [Table 9](#).

Field	Description
pgx.baseUrl	Enter the PGX Base URL. This is the location where the data is pushed. For example, <code>http://<HOSTNAME>:7007</code>
default.url	Enter the jdbc URL. NOTE: If you want to use the Oracle wallet credentials, you must enter the alias name in the following format: <code>jdbc:oracle:thin:@<alias_name></code>
zeppelin.jdbc.principal	Enter the principal name to load from the keytab file.
default.driver	Enter the default JDBC driver name.
default.completer.ttlInSeconds	Enter the time to live SQL completer in seconds.
default.password	Enter the default password. NOTE: This value can be null if you have entered the alias name in the <code>default.url</code> parameter for the jdbc interpreter.
default.splitQueries	Each query is executed apart and returns the result. Specify the presence of default split queries. Enter 'True' to split or 'False' not to.
default.completer.schemaFilters	Enter comma-separated schema filters to get metadata for completions.
ofsaa.sessionservice.url	Enter the session service URL. For example, <code>http://<HOSTNAME>:7047/sessionservice</code> Here, <HOSTNAME> refers to the server name or IP address where fcc-studio is installed.

default.user	Enter the name of the default user.
zeppelin.jdbc.concurrent.max_connection	Enter the number of maximum connections allowed.
ofsaa.metaservice.url	Enter the metaservice URL. For example, <code>http://<HOSTNAME>:7045/metaservice</code> Here, <HOSTNAME> refers to the server name or IP address where fcc-studio is installed.
common.max_count	Enter the maximum number of SQL result to display.
zeppelin.jdbc.auth.type	Enter the default JDBC authentication type. The authentication methods supported are SIMPLE and KERBEROS
zeppelin.jdbc.precode	Enter the snippet of code that executes after the initialization of the interpreter.
zeppelin.jdbc.concurrent.use	Specify concurrent use of JDBC connections. Enter 'True' to enable or 'False' to disable.
zeppelin.jdbc.keytab.location	Enter the keytab file location.

4.1.7.2 Link Wallet Credentials to jdbc Interpreter

FCC Studio provides secure and safe credential management. Examples for credentials are passwords, Oracle Wallets, or KeyStores. Use this section to link credentials (a wallet and a password) to jdbc interpreter variant to enable secure data access. This linking enables the jdbc interpreter to securely connect to the specified Oracle database. For more information to link Wallet Credentials to jdbc Interpreter, see [Link Credentials](#).

NOTE The Credentials' section is enabled if an interpreter variant can accept credentials.

You can also create new credentials and link to jdbc Interpreter. For more information, see [Create Credentials](#).

4.1.8 md Interpreter

This interpreter is used to display text based on Markdown, which is a lightweight markup language. In the md interpreter, you can configure the markdown parser type. Markdown (md) is a plain text formatting syntax designed so that it can be converted to HTML. Use this section to configure the markdown parser type.

To configure the md interpreter variant, follow these steps:

1. On the md Interpreter page LHS menu, select md. The md interpreter pane is displayed.

2. Enter the markdown parser type and click Update. To confirm the modified configuration.

4.1.9 pgql Interpreter

The `pgql` Interpreter is a ready-to-use interpreter used to connect the configured PGX server. This interpreter is used to perform queries on the graph in FCC Studio. In the `pgql` Interpreter, you can configure the class which implements the formatting of the visualization the output, the size of the output message, and so on.

PGQL is a graph query language built on top of SQL, bringing graph pattern matching capabilities to existing SQL users and to new users who are interested in graph technology but who do not have an SQL background.

To configure the `pgql` interpreter variant, follow these steps:

1. On the Interpreter page LHS menu, select `pgql`. The `pgql` interpreter pane is displayed.
2. Enter the following information in the `pgql` interpreter variant pane as tabulated in [Table 10](#).

Field	Description
<code>graphviz.formatter.class</code>	Enter the class which implements the formatting of the visualization output. For example, <code>oracle.datastudio.graphviz.formatter.DataStudioFormatter</code>
<code>graphviz.driver.class</code>	Enter the class which implements the PGQL driver. For example: <code>oracle.pgx.graphviz.driver.PgxDriver</code>
<code>base_url</code>	Enter the base URL of the PGX. For example, <code>http://<HOSTNAME>:7007</code>
<code>zeppelin.interpreter.output.limit</code>	Enter the output message limit. Any message that exceeds the limit is truncated. For example, 102 or 400.

4.1.10 pgx-algorithm Interpreter

The `pgx-algorithm` Interpreter is a ready-to-use interpreter used to connect to the configured PGX server. This interpreter is used to write an algorithm on the graph, and it is also used in the `pgx-java` interpreter. In the `pgx-algorithm` Interpreter, you can configure the class which implements the PGQL driver, the size of the output message, and so on.

To configure `pgx-algorithm` interpreter variant, follow these steps:

1. On the Interpreter page LHS menu, select `pgx-algorithm`. The `pgx-algorithm` interpreter pane is displayed.
2. Enter the following information in the `pgx-algorithm` interpreter variant pane as tabulated in [Table 11](#).

Field	Description
<code>graphviz.formatter.class</code>	Enter the class which implements the formatting of the visualization output. For example, <code>oracle.datastudio.graphviz.formatter.DataStudioFormatter</code>
<code>graphviz.driver.class</code>	Enter the class which implements the PGQL driver. For example, <code>oracle.pgx.graphviz.driver.PgxDriver</code>
<code>base_url</code>	Enter the base URL of the PGX server.

4.1.11 `pgx-java` Interpreter

The `pgx-java` Interpreter is a ready-to-use interpreter used to connect to the configured PGX server. This interpreter is used to write an algorithm on the graph and also used in the `pgx-java` interpreter. In the `pgx-java` Interpreter, you can configure the class which implements the formatting of the visualization output, the class which implements the PGQL driver, and so on.

`PGX-java` interpreter is Java11 based interpreter with PGX client embedded in it to query on graph present in the PGX server.

To configure the `pgx-java` interpreter variant, follow these steps:

1. On the Interpreter page LHS menu, select `pgx-java`. The `pgx-java` interpreter pane is displayed.
2. Enter the following information in the `pgx-java` interpreter variant pane as tabulated in [Table 12](#).

Field	Description
<code>graphviz.formatter.class</code>	Enter the class which implements the formatting of the visualization output. For example, <code>oracle.datastudio.graphviz.formatter.DataStudioFormatter</code>
<code>graphviz.driver.class</code>	Enter the class which implements the PGQL driver. For example, <code>oracle.pgx.graphviz.driver.PgxDriver</code>

base_url	Enter the base URL of the PGX server in this field.
zeppelin.interpreter.output.limit	Enter that the output message from the interpreter exceeding the limit will be truncated. For example, 102,400.

4.1.12 pyspark Interpreter

Users must write for connection either in the Initialization section or in the notebook's paragraph. This interpreter is used to write the pyspark language to query and perform analytics on data present in big data. This requires additional configuration, which must be performed as a prerequisite or as post-installation with the manual change of interpreter settings.

In the pyspark interpreter, you can configure the Python binary executable to use for PySpark in both driver and workers, set 'True' to use IPython, else set to 'False', and so on.

To configure the pyspark interpreter variant, follow these steps:

1. On the Interpreter page LHS menu, select pyspark. The pyspark interpreter pane is displayed.
2. Enter the following information in the pyspark interpreter variant pane as tabulated in [Table 13](#).

Field	Description
zeppelin.pyspark.python	Enter the Python binary executable to use for PySpark in both drivers and workers. The default value is python. For example, <code>python</code>
zeppelin.pyspark.useIPython	Set to 'True' to use IPython, else set to 'False'.

4.1.13 spark Interpreter

The spark Interpreter does not connect to any schema by default. Users must write for connection either in the Initialization section or in a notebook's paragraph. This interpreter is used to perform analytics on data present in Big data clusters in the Scala language. This requires additional configuration, which must be performed as pre-requisite or as post-installation with the manual change of interpreter settings.

In spark interpreter, you can configure the cluster manager to connect, print the Read-eval-print loop (REPL) output, the total number of cores to use, and so on.

To configure the spark interpreter variant, follow these steps:

1. On the Interpreter page LHS menu, select spark. The spark interpreter pane is displayed.
2. Enter the following information in the spark interpreter variant pane as tabulated in [Table 14](#).

Field	Description
pgx.baseUrl	Enter the PGX Base URL. This is the location where the data is pushed. For example, <code>http://<HOSTNAME>:7007</code>
spark.executor.memory	Enter the amount of memory to use for the executor process. Executor memory per worker instance. For example, 512m and 32g. In Spark, the executor-memory flag controls the executor heap size (similarly for YARN and Slurm), the default value is 512MB per executor. And the driver-memory flag controls the amount of memory to allocate for a driver, which is 1GB by default and should be increased in case you call a collect or take (N) action on a large RDD inside your application.
spark.master	Enter the cluster manager to connect to. For example, <code>local[*]</code>
spark.yarn.archive	Enter the archive containing the required Spark jars for distribution to the YARN cache, to make Spark runtime jars accessible from the YARN side.
spark.app.name	Enter the name of the application. For example, <code>Zeppelin</code>
zeppelin.spark.ui.hidden	Set to True or False.
zeppelin.spark.maxResult	Enter the maximum number of results that must be fetched.
spark.pyspark.python	Enter the Python binary executable to use for PySpark in both driver and executors. For example, <code>python</code>
zeppelin.spark.enableSupportedVersionCheck	Set to 'True' or 'False'.
args	Enter the Spark command-line args.
zeppelin.spark.useNew	Set to 'True' to use the new version of the SparkInterpreter.
zeppelin.spark.useHiveContext	Set to 'True' to use HiveContext instead of SQLContext.

zeppelin.spark.uiWebUrl	Overrides Spark UI default URL. Value should be a full URL (<code>http://{hostName}/{uniquePath}</code>)
zeppelin.spark.printREPLOutput	Enter to print the REPL output.
spark.cores.max	Enter the total number of cores to use. NOTE: Empty value uses all available cores.

4.2 Link Credentials

FCC Studio provides secure and safe credential management. Examples for credentials are passwords, Oracle Wallets, or KeyStores. Use this section to link credentials (a wallet and a password) to fcc-jdbc or jdbc interpreter variant to enable secure data access. This linking enables the fcc-jdbc or jdbc interpreter to securely connect to the specified Oracle Database. You can also create new credentials, based on your requirement to connect to the new interpreter variants. For more information, see [Create Credentials](#).

NOTE You can link credentials only to fcc-jdbc and jdbc interpreters. The Credentials' section is enabled if an Interpreter variant can accept credentials.

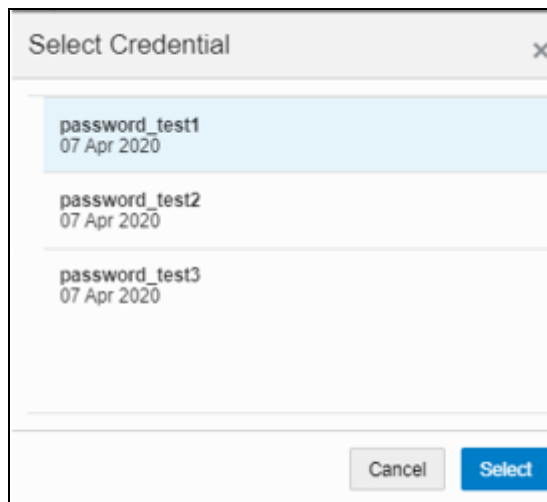
To link ready-to-use credentials to the required interpreters, follow these steps:

1. On the Interpreters page, select the required interpreters. For example, fcc-jdb or jdbc.
2. Go to the **Credentials** section.

3. To select Oracle Wallet (jdbc wallet) credential that you want to link to the Interpreter variant, click Select. The Select Credential dialog is displayed.



4. Select the required Oracle Wallet (jdbc wallet).
5. To select Password (jdbc password) that you want to link to the Interpreter variant, click Select. The Select Credential dialog is displayed.



6. Select the required Password (jdbc password). Click Select.
7. Click Update to save the changes. The required password and Oracle Wallet are linked to the fcc-jdbc or jdbc Interpreter.

4.3 Create a Credential

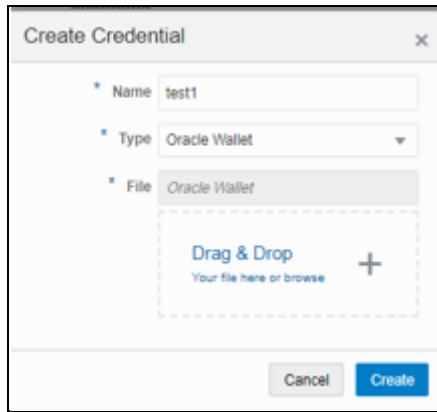
New credentials are created when database details are changed or updated. For example, change in Transparent Network Substrate (TNS) due to hostname change or compulsory periodic update of passwords of the schema.

Oracle Wallet provides a simple and easy method to manage database credentials across multiple domains. It allows you to update database credentials by updating the Wallet instead of having to change individual data source definitions.

Use this section to add a new credential to the interpreters.

To create a credential, follow these steps:

1. On the FCC Studio workspace LHS Menu, click Credentials. The Credentials page is displayed.
2. Click Create. The Create Credential dialog is displayed.



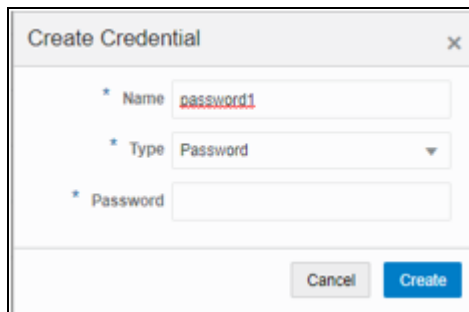
3. Enter the following information in the Create Credential dialog box as tabulated in [Table 15](#):

Field	Description
Name	Enter the name for the wallet credential.
Type	Select Oracle Wallet.
File	<p>Upload the wallet zip file that includes the following files:</p> <ul style="list-style-type: none"> • cwallet.sso • ewallet.p12 • tnsnames.ora <p>NOTE:</p> <ul style="list-style-type: none"> • The wallet file must be in .zip format. • The maximum file size allowed for the credential file is 128Kb.

4. Click Create. The wallet credential is created and displayed on the Credentials page.


To create a new password credential for the wallet, follow these steps:

1. Click Create. The Create Credential dialog is displayed.



2. Enter the following information in the Create Credential dialog as tabulated in [Table 16](#).


Field	Description
Name	Enter the name for the password credential.
Type	Select password type from the drop-down (wallet or keytab).
Password	Enter the wallet password for the password credential.

3. Click Create. The password is created for the wallet and displayed on the Credentials page.
4. To download the credential files, click the credential file name on the Credentials page.
5. To delete a required credential, click Delete . The credential is removed from the list.

4.4 Create an Interpreter Variant

In FCC Studio, you can either use a default interpreter variant or create a new variant for an interpreter. You can create more than one variant for an interpreter. Multiple variants for an interpreter are created to connect different versions of interpreters (Python ver: 3, Python ver: 2), connect a different set of users, database schema. For example, FCC Studio schema, BD schema, and so on.

To create a new interpreter variant, follow these steps:

1. On the Interpreters page, click the required interpreters from the LHS list. For example, fcc-jdbc interpreter.
2. The default interpreter variant is displayed on the RHS.
3. On the default interpreter, click Add  to create a new variant. The Create Interpreter Variant dialog box is displayed.
4. Enter the Name for the new interpreter variant. Click Create. A new variant is created with a name, <Interpreter Type>.<Variant Name>.
5. Provide the new schema details such as the default.url, default.user, and default.password.

NOTE Steps: 5 and 6 are applicable only for fcc-jdbc

6. The Oracle Database schema that you have created must be granted with the same permissions that are granted to the BD or ECM atomic schema.

7. For more information, see the Prerequisite Environmental Settings section in the [OFS Crime and Compliance Studio Installation Guide \(On-Premise\)](#).
8. Navigate to the <Studio_Installation_Path>/ficdb/bin directory. Run the following 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
```
9. Configure the required values for the properties.
10. Click Update. A new variant is created for the selected interpreter.
11. For using the new interpreter variant in the notebook paragraphs, use the following format:
12. %fcc-jdbc.<VariantName>

4.5 Enable Second Spark or PySpark interpreter

Interpreter variants do not apply to Spark or PySpark interpreter. Hence, you must enable an additional set of interpreters.

To enable a second Spark or PySpark interpreter, see Enabling a Second Spark or PySpark Interpreter chapter in the [OFS Crime and Compliance Studio Installation Guide \(On-Premise\)](#).

4.6 Modify the Python Docker Images for the Python Interpreter

Use this section to modify the Python Docker images in the Kubernetes environment. A Docker image is built by a series of layers. Each layer represents an instruction in the image's Docker file. Each layer except the very last one is read-only.

When the FCC studio is installed and started, the image is loaded on the local node and pushed into a docker repository. The image can be modified on the local node or any machine which can pull in and push to the Docker repository.

To modify the Python packages or change the Python version, you must modify the Python image.

NOTE

- This section is applicable for FCC Studio installed using Kubernetes.
- This section can be used as an example to understand the steps involved to modify the Python images for the Python interpreter in FCC Studio.

Topics:

- [Prerequisite to Build a Python Interpreter Docker Image](#)
- [Build and Pushing an Image](#)
- [Replace Python Image in FCC Studio](#)

4.6.1 Prerequisite to Build a Python Interpreter Docker Image

To build Python images, you can either modify the Python packages in the Python interpreter or add different versions of Python to the Python interpreter.

4.6.1.1.1 Modify Python Packages

Python packages are present inside the python interpreter image, FCC Studio allows you to modify the version or upgrade to a new Python package in the Python interpreter.

The following Python libraries are part of the fcc-python interpreter images for Python 3.6 version:

- pandas 0.25.3
- numpy 1.17.4
- scipy 1.3.2
- scikit-learn 0.21.3
- matplotlib 3.1.1
- seaborn 0.9.0
- cx-oracle 7.2.2
- sqlalchemy 1.3.11

To modify the Python packages in the Python interpreter, follow these steps:

NOTE This process adds the Python packages to Python 3.6.

1. Navigate to the Studio Installation Path directory.
2. Create a directory in the same location as the `<Studio_Installation_Path>` and create a file inside the directory as `Dockerfile`.
3. Copy and paste the following information as a template into the `Dockerfile`:
4. `FROM ofsaa-fccm-docker-release-local.dockerhub-den.oraclecorp.com/fcc-studio/fcc-python:8.0.8.0.0`
5. `RUN pip3 --no-cache-dir install scipy pandas cx_oracle --user`
6. Modify the `Dockerfile` depending on the following installation method:
 - a. If Internet connectivity is available, follow these steps:
 - b. Depending on the version of the Python package, install the `scipy` and `cx_oracle` Python packages using the following command:

- c. `RUN pip install scipy cx_oracle`
- d. If Internet connectivity is unavailable, follow these steps:
 - i. Download the Python package files.
 - ii. Create a directory besides the `Dockerfile` file.
 - iii. For example, `packages`
 - iv. Place the downloaded files in the newly created `packages` directory.
 - v. Modify the `Dockerfile` using the following commands:
 - vi. For example, to install using Python3:
 - vii. `COPY packages /var/olds-python-interpreter/packages`
 - viii. `RUN cd /var/olds-python-interpreter/packages`
 - ix. `RUN pip3 --no-cache-dir install --no-deps numpy-1.17.4-cp36-cp36m-manylinux1_x-86_64.whl`

NOTE

For more information on how to write `Dockerfile`, visit <https://docs.docker.com/engine/reference/builder/>.

7. Build and push the image to the Docker registry. For more information, see [Build and Push an Image](#)
8. Replace the Python image in FCC Studio. For more information, see [Replace a Python Image in FCC Studio](#)

4.6.1.1.2 Add Different Version of Python

Python packages are present inside the python interpreter image, FCC Studio allows you to add a new version or upgrade to a new Python package in the Python interpreter. Use this section to add different versions or upgrade to a new Python version to a Python interpreter.

To add a different version of Python to Python interpreter in FCC Studio, follow these steps:

1. Navigate to the Studio Installation Path directory.
2. Create a directory at the same location as the `<Studio_Installation_Path>` and create a file inside the directory with the file name, `Dockerfile`.
3. Copy and paste the following information as a template into the `Dockerfile`:

```
FROM ofsaa-fccm-docker-release-local.dockerhub-  
den.oraclecorp.com/fccstudio/ fcc-  
python:8.0.8.0.0  
USER root  
RUN yum install -y python3.5  
USER interpreteruser
```

```
RUN python3.5 -m pip install scipy pandas cx_oracle -
-user
```

4. Modify the `Dockerfile` based on the preferred way of installing Python in the RHEL server.

For more information on how to modify the `Dockerfile`, visit

<https://docs.docker.com/engine/reference/builder/>.

5. Build on LINUX and push the image to the Docker Registry. For more information, see [Build and Push an Image](#)
6. Replace the Python image in FCC Studio. For more information, see [Replace a Python Image in FCC Studio](#)

4.6.2 Build and Push an Image

An image is built on LINUX and pushed to a docker Registry.

To build and push an image, follow these steps:

1. Navigate to the Studio Installation Path directory.
2. Build the docker image using the following command:

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

Where:

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

For example,

```
docker build . --build-arg http_proxy=http://my-proxy-
url:80 --build-arg https_proxy=http://my-proxy-url:80 -t
my.docker-registry.com:5000/ofsaa-fccm-docker-release-
local.dockerhub-den.oraclecorp.com/fcc-studio/fcc-
python:8.0.8.0.0-C1
```

NOTE

`build-arg` can be skipped if proxy is not required or if packages are placed locally.

3. Push the images using the following command:

```
docker push <my.docker-registry.com:port>/ofsaa-fccm-
docker-releaselocal.dockerhub-den.oraclecorp.com/fcc-
studio/fcc-python:<version>
```

4.6.2.1 Replace a Python Image in FCC Studio

Python packages are present inside the python interpreter image, but if you want to replace the Python images, FCC Studio allows you to perform this activity. Use this section to replace a Python image in the FCC Studio.

To replace the Python images in FCC Studio, follow these steps:

1. Navigate to the `<Studio_Installation_path>/deployments/` directory.
2. Update the image name in the `fcc-python.yml` file as follows:

```
spec:
  spec:
    containers:
      - name: python-interpreter
        image: ofsaa-
fccm-docker-release-local.dockerhub-den.oraclecorp.com/fcc-
studio/fcc-python:<version>
```

3. To restart the FCC Studio application, follow these steps:
 - a. Execute the following command from the Kubernetes master node:
`kubectl delete namespace <Namespace>`
 - b. Navigate to the `<Studio_Installation_Path>/bin` directory.
 - c. Execute the following command:
`./fcc-studio.sh --registry <registry URL>:<registry port>`

5 Prepare the Batches

Use this section to prepare batches to execute the ETL. Batches enable you to load graphs, run notebooks, and move data from Oracle Database or Big Data to FCC Studio. Batches are prepared based on the Realms you are using.

- [Prepare Batches for FCCRealm](#)
- [Prepare Batches for FCCSAML Realm](#)

5.1 Prepare Batches for FCCRealm

To prepare the batches for FCCRealm, follow these steps:

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

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

NOTE

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

5.2 Prepare Batches for FCCSAML Realm

To prepare the batches for FCCSamlRealm, you must generate an API token and configure the `NBExecutor.properties`.

To generate the API token, follow these steps:

1. Navigate to the `<Studio_Installation_Path>/ficdb/bin` directory.
2. Run the shell script: `fic_db_home/FCCM_Studio_Generate_APIToken.sh`.
3. Run the script: `fic_db_home/FCCM_Studio_Generate_APIToken.sh`
APPNAME.

For example, use the `./FCCM_Studio_Generate_APIToken.sh`
`BATCH_USER`.

In the `NBExecutor.properties` file, specify the following details:

- `saml=true`
- `username=<BATCH_USERNAME>`, blank if **SAML** is **true**.
- `password=<BATCHUSER_PASSWORD>`, blank if **SAML** is **true**.
- `apiToken=<API_TOKEN>`

NOTE `BATCH_USERNAME` and `BATCHUSER_PASSWORD` can be NULL.

6 Execute Published Notebook

A notebook is a collection of documentation and snippets of executable code. The notebook allows large scripts to be broken into a modular collection of executable code with tailored results. Different languages, such as Groovy, Scala, Python, and Oracle's own property graph query language (PGQL), can be combined into one notebook. Each notebook is mapped to the role of the logged-in user.

A notebook is modifiable by the author.

When a notebook is published:

- The original notebook is cloned, and a published notebook is created.
- Any changes made to the original notebook will have no impact on the published notebook.
- Whenever the original notebook is re-published, a new version of the published notebook is created.
- The published notebook is in a read-only format.
- The published notebook can be run in a batch pipeline.

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

To execute the published notebook, follow these steps:

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

6.1 Execute Published Scenario Notebook for FCC Studio with Non-OFSAA

The scenario notebook is of Notebook Type, Scenario. For scenario notebooks, the publish functionality provides four eyes approval process. When a scenario notebook is published, the published notebook is shared with the user who is mapped to the DSBATCHGRP group for approval.

When a notebook is approved by the batch user:

- The original notebook is cloned, and a published notebook is created.
- Scenario metadata is created.

- A threshold set is created with values for all the parameters that you have provided in the notebook.

The approved scenario notebook can be executed with any threshold set which is created when the notebook is being approved by the batch user. Whenever a user wants to create a new threshold set, the notebook must be re-published with different threshold values. After a notebook is approved, a new threshold set is created for the same scenario.

NOTE

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

You can obtain the **notebookID** by manually extracting it from the URL when the notebook is open in the UI. The format of the notebookID is **dsXXXXXX**.

URL: <FCC studio
URL>/?root=notebooks¬ebook=ds2gbW38

For example, **ds2gbW38** is notebookID.

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

1. Navigate to the <Studio_Installation_Path>/ficdb/bin directory.
2. Set `FIC_DB_HOME` as an environment variable.
3. Execute the following command with proper notebookID for batch execution of notebook.

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

6.2 Execute Published Non-Scenario Notebook for FCC Studio with Non-OFSAA

The non-scenario notebook is of Notebook Type, Default, or Jupyter. Upon publishing an original notebook, you must select the user role or group to which the notebook will be published. After publishing, the original notebook is cloned, and a published notebook is created.

NOTE

- This section is applicable for FCC Studio with non-OFSAA.
- The **FCCM_Studio_NotebookExecution** script requires the **notebookID** parameter.

You can obtain the **notebookID** by manually extracting it from the URL when the notebook is open in the UI. The format of the notebookID is **dsXXXXXX**.


```
URL: <FCC studio
URL>/?root=notebooks&notebook=ds2gbW38
```

For example, **ds2gbW38** is notebookID.

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

1. Navigate to the <Studio_Installation_Path>/ficdb/bin directory.
2. Set FIC_DB_HOME as an environment variable.

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

For example, If

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

Then the extraparams must be written as follows:

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

7

Enable Synonym and Stopword with the Elastic Search Service

To enable Synonym and Stopword with the Elastic Search service, follow these steps:

1. Navigate to the `<Elastic_Search_Installed_Path>/config` directory.
2. Create a directory named `analysis` using the following command:

```
mkdir analysis
```
3. Place your Stopword and Synonym files in the newly created `analysis` directory.

Some examples are provided here:

NOTE

- User can decide to provide any data in the Stopword or Synonym files.
 - Each stopword must be provided in a separate line.
 - All related synonyms must be provided in the same line separated by a comma.
 - All the synonyms must be provided in the same line and ensure that there are no repetitions of the synonym. For Example: `rob, robi, robie, roby, robbi`.
-
- `Synonyms.txt`: Contains name synonyms like `bob, bobby`, and so on.
 - `Country.txt`: Contains country synonyms like `US, United States, America`, and so on.
 - `Organisation_suffix.txt`: Contains organization suffices that are used as stopwords.
 - `Title.txt`: Contains title stopwords used as the title for name.
 - `Gender.txt`: Contains gender-related synonyms.
 - `Organisation_strip.txt`: Contains organization stopwords.

8 Configure ETL

Use this chapter to understand and perform configuration activities before running the extract, transform, and load (ETL) process. The ETL process loads business data from FCDM (BD and ECM) which can be used by any interpreter. FCDM and External data sources such as ICIJ are processed and loaded as a graph in FCC Studio.

Topics:

- [Understand ETL](#)
- [Configure RuleSet](#)
- [Configure Data Source](#)
- [Configure Graph](#)
- [Apply Graph Fine-Grained Access Control](#)

8.1 Understand ETL

Use this section to understand how to move source data into the PGX server to generate a graph. The following sections provide more insight on data sources, jobs, rulesets, and workflows used in FCC studio to generate graphs.

Topics:

- [Data Source](#)
- [Rulesets](#)
- [Indices](#)
- [Elastic Search](#)
- [PGX](#)
- [ETL and its Workflow](#)
- [Graph Model](#)

8.1.1 Data Source

The FCC Studio provides the following ready-to-use data sources:

- FCDM: The Financial Crime Data Model (FCDM) is the data source for Behavior Detection (BD) and Enterprise Case Management (ECM) Atomic schema tables.
- ICIJ: International Consortium of Investigative Journalists (ICIJ) is the data source for external entities like Panama Papers, Paradise Papers, and so on. Where input data files are in the .CSV format and these files must be placed in Hadoop Distributed File System (HDFS).

8.1.2 Rulesets

Ruleset facilitates to identify the similarity between two entities within the data source that is, FCDM to FCDM (customer to customer) or across the data source that is, FCDM to ICIJ

(FCDM entities, customer to customer, and customer to Panama papers) to derive a match and create similarity edges in the graph. A Ruleset is a set of rules that are applied to the defined source and target entities, compares the attributes of the entities to derive a match for similarity edges. FCC studio provides ready-to-use rulesets; however, you can modify these rulesets or create your rulesets.

Based on the rulesets, a task is executed. The indices perform the correlation and based on the threshold score and weightage the similarity edges are created. The threshold scoring parameters such as Automatic Threshold and Manual Threshold determine the creation of similarity edges.

- Auto Threshold: Enter values from 0 to ≤ 1 . Enter a matching percentage above which you want the matches to get into the graph automatically.
- Manual Threshold: Enter values from 0 to ≤ 1 . Enter a value above which you want to do borderline decisions of similarity edges. It is considered as a range greater than the Manual threshold and less than the Automatic threshold.

The ready-to-use threshold scoring logic works as explained in the following example:

- A total threshold score of less than 0.7 (70%) must be discarded.
- A total threshold score of more than or equal to 0.8 (80%) must create a similarity match automatically in the graph. This is configured using the Automatic Threshold parameter in the Ruleset.
- A total threshold score between 0.7 (70%) and less than 0.8 (80%) must be decided manually by the user to publish the similarity matches in the graph. This is configured using the Manual Threshold parameter in the Ruleset.

Users can increase the threshold values based on false positive matches. If there are more false positives, it means the threshold value is less. To get more positive matches, increase the threshold value.

8.1.3 ElasticSearch

An Elasticsearch is a distributed search and analytical engine for all types of structured and unstructured data. In FCC Studio, Node tables are moved to Elasticsearch to get faster responses to identify the entity's relationships.

8.1.4 Indices

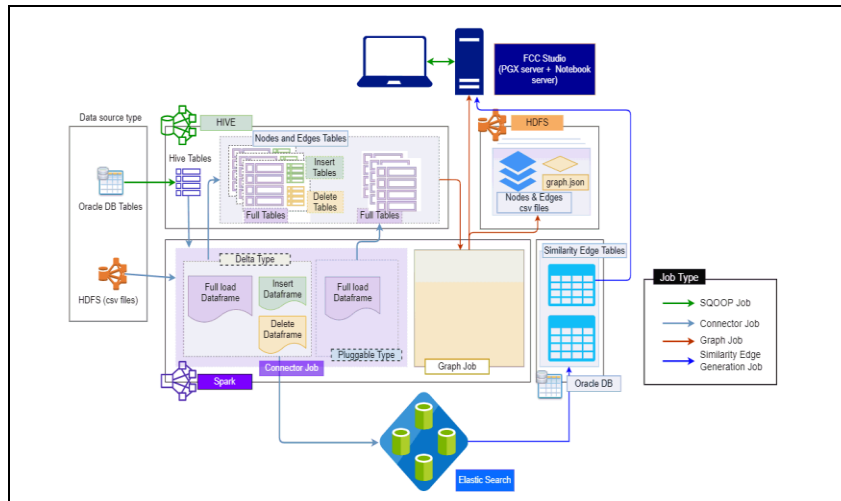
An index is a logical namespace that maps to one or more primary shards (a shard is a unit in which Elasticsearch distributes data around the cluster). As a part of the connector's job, the node entities from the graph's data source are populated as indices. The indices are used to query to generate similarity edges in the graph.

8.1.5 PGX

PGX is a toolkit for graph analysis, supporting both efficient graph algorithms and fast SQL-like graph pattern matching queries.

8.1.6 ETL and its Workflow

FCC Studio's ETL is a process, where the business data and external data can be processed to get a global graph, with entity resolved based on matching rules. This graph can be further queried for investigation.



ETL has two stage:

- Generation and maintenance of Graph
- Resolving entities based on matching rules and updating

In first stage, the business data and external data are collected in HIVE Schema and then transformed based on query and saved into HIVE tables. The Node type data are updated in elastic search for second stage of ETL.

ETL classifies entity into “delta type” or “pluggable type” and based on entity type it updates the change, referred as delta, into graph. For delta type, the ETL compares data from previous batch and recognizes changes as insert or delete.

Example: To understand the ETL delta, see 14 Appendix Example of ETL Delta.

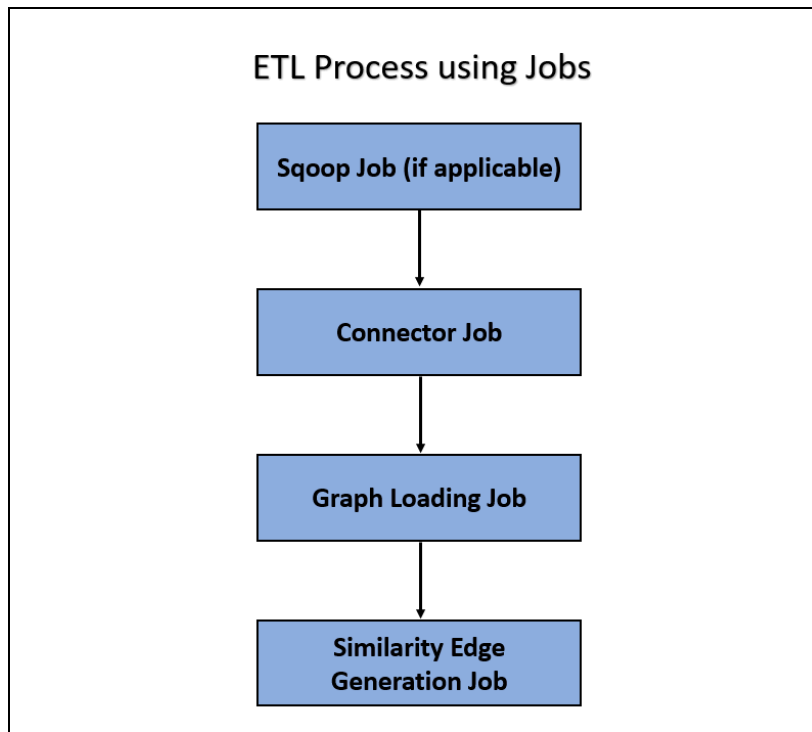
For pluggable type which are transaction edges, the data are separated into multiple datasets based on a parameter like ‘transaction date’ and then are updated into graph based on the valid range of transaction date. ETL also removes older transaction and thus maintains graphs with desired range of transactions.

In second stage, based on matching rules (ruleset) the nodes are resolved and similarity edges are generated among which edges having similarity score more than automatic thresholds are directly updated into graph and other edges with score between manual and automatic threshold can be review from FCC Studio by users and as soon as they are approved, these edge(s) are updated into graph.

8.1.7 Jobs

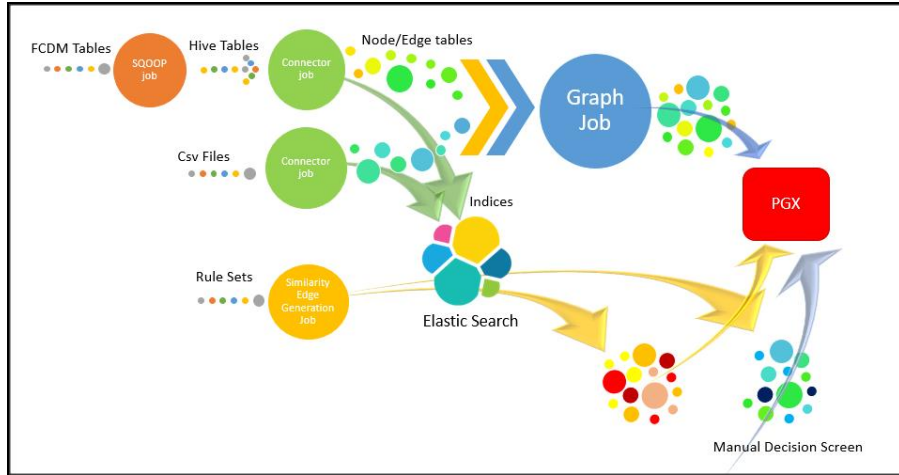
The ETL process is split into four Jobs: Sqoop Job (if applicable), Connector Job, Graph Loading, and Similarity Edge Generation Job.

The following image illustrates the sequence of the ETL process.



- **Sqoop Job:** This task moves data from Behavior Detection (BD) and Enterprise Case Management (ECM) Atomic tables to Hive tables based on the date range. The Sqoop job creates and saves the import and export commands. It specifies parameters to identify and recall the saved job. This recalling or re-executing is used in the incremental import, which can import the incremental data from the RDBMS table to Hive.
- **Connector Job:** This task transforms data from Hive tables or the .csv files into nodes and edges format and identifies the changes in data loaded between previous and current batch. This task also pushes node tables into Elastic Search.
- **Graph Job:** This task generates the .csv files and configuration files for the graph, update the changes into a graph, and manages transaction edges as per the date range.
- **Similarity Edge Generation Job:** This task generates the similarity edges based on a ruleset in the FCC Studio application and adds similarity edges for breaching automatic threshold into the graph directly.

The following image illustrates the workflow of ETL for specific datasources such as, Ready-to-use datasource, FCDM, and ICIJ:



For more information on Jobs execution, see [Run ETL](#).

To understand how to move source data to the PGX server using connector jobs to create graphs in FCDM and ICIJ workflows as tabulated in [Table 17](#).

Jobs	FCDM Workflow	ICIJ Workflow
Sqoop Job	Moves data from FCDM (BD or ECM) source into Hive tables.	Not applicable
Connector job	<ul style="list-style-type: none"> Transforms data into nodes and edges tables using ready-to-use queries Identifies incremental and updated data Pushes nodes into Elasticsearch as indices 	<ul style="list-style-type: none"> Reads the .csv files and transforms the data into nodes and edges tables using ready-to-use queries Identifies incremental and updated data Pushes nodes into Elasticsearch as indices
Graph Loading Job	<ul style="list-style-type: none"> Generates the .csv files and configuration files to load graph into the PGX server Loads the delta graph changes 	<ul style="list-style-type: none"> Generates the .csv files and configuration files to load graph into the PGX server Loads the changes directly into the PGX server from subsequent batches Loads the delta graph changes
Similarity edge Generation Job	<ul style="list-style-type: none"> Generates similarity edges Pushes automatic matches of similarity edges into graph 	<ul style="list-style-type: none"> Generates similarity edges Pushes automatic section of similarity edges into graph

After running the ETL process, global graphs are generated. For more information on the ready-to-use Graph Model, see [Graph Model](#).

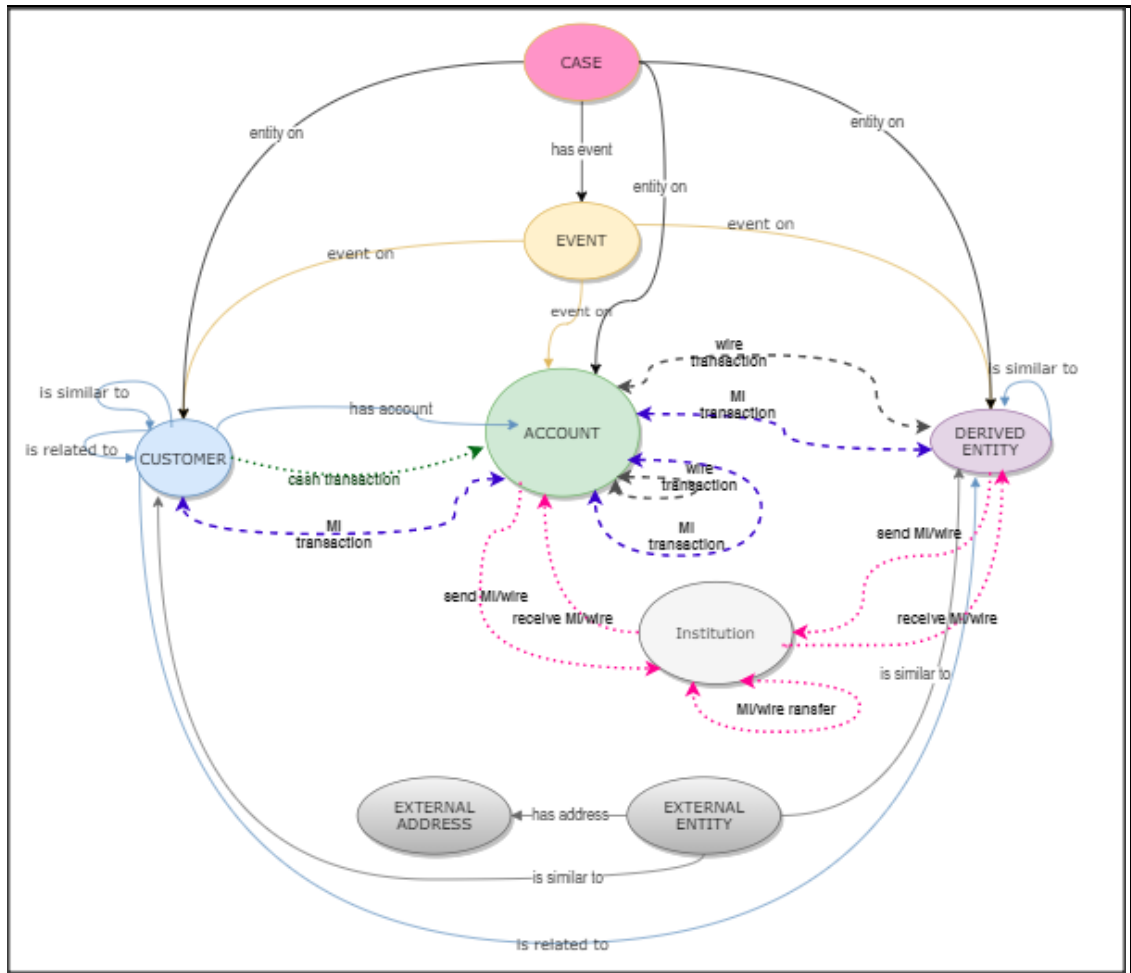
For more information on Jobs execution, see [Run ETL](#).

8.1.8 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. The graph model expresses the conditional dependence structure between nodes and edges.

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

The following image illustrates the Graph Model.



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

8.2 Configure Rulesets

In FCC Studio, data is obtained from FCDM (Financial Crime Data Model) to generate Financial Crime Graph Model. The graph model includes nodes for entities such as

Customers, Accounts, Events, and Derived Entities, and edges for transactions and relationships.

Entity Resolution compares nodes to identify pairs or groups of nodes that refer to the same entity. Entity Resolution creates Similarity Edges between nodes by comparing the attributes of the nodes and identifying where the similarity is significant enough to create an edge so the nodes are linked with the graph model and can be analyzed as a single entity.

Entity matching rules are used to compare nodes of different types. For example, deduplicating customers, resolving derived entities, linking customers or derived entities to external data such as Panama papers or sanctions lists with different rules and thresholds. For more information on matching rulesets, see the [Financial Crime Graph Model Matching Guide](#).

The ruleset facilitates to identify the similarity between two entities (customer, account, and so on) and derive a match. A Ruleset is a set of rules that are applied to the defined source and target entities, compares the attributes of the entities to derive a match. Rulesets are configured before running the ETL to move the data from the source database (BD atomic schema) into Hive schema. Graphs are also configured for similarity edges to identify the similarity between two entities (customers, account, and so on). Entity details and attributes are configured in the rulesets when there any modification or addition to the entity details such as Email ID, Tax ID, or jurisdiction. Similarly, when there are changes to thresholds or scoring methods rulesets are configured.

FCC studio provides ready-to-use rulesets; however, you can modify these rulesets or create your rulesets.

Topics:

- [Use Rulesets](#)
- [Create Rulesets](#)
- [Create Rules in a Ruleset](#)
- [Scoring Method](#)

8.2.1 Use Rulesets

Use this section to enable or disable the ready-to-use rulesets. You can also modify source or target entities, attributes for these rulesets, and also remove rulesets from the existing list.


To manage ready-to-use rulesets, follow these steps:

1. In the Crime and Compliance Studio LHS Menu, select Ruleset. The Ruleset page is displayed with all ready-to-use rulesets.

Figure 1: Ruleset Page

RULESET NAME	DESCRIPTION	AGGREGATION	SOURCE ENTITY	TARGET ENTITY	ENABLE
Customer To Ext Source - Paradise Addr	Match Customer Address To External Source - Paradise Address	Maximum	customer	external_address_paradise	<input type="checkbox"/>
Customer To Ext Source - Panama	Customer To External Source - Panama	Maximum	customer	external_entity_panama	<input type="checkbox"/>
Customer To Customer Match		Maximum	customer	customer	<input type="checkbox"/>
Customer To Ext Source - Panama Addr	Match Customer Address To External Source - Panama Address	Maximum	customer	external_address_panama	<input type="checkbox"/>
Customer To Derived Entity	Customer To Derived Entity	Maximum	customer	derived_entity	<input type="checkbox"/>
Derived Entity To Derived Entity	Derived Entity To derived Entity Match	Maximum	derived_entity	derived_entity	<input type="checkbox"/>
Customer To Ext Source - Offshore	Match Customer To External Source - Offshore	Maximum	customer	external_entity_offshore	<input type="checkbox"/>
Customer To Ext Source - Bahamas	Match Customer To External Source - Bahamas	Maximum	customer	external_entity_bahamas	<input type="checkbox"/>
Customer To Ext Source - Bahamas Addr	Match Customer Address To External Source - Bahamas Address	Maximum	customer	external_address_bahamas	<input type="checkbox"/>
Customer To Ext Source - Paradise	Match Customer To External Source - Paradise	Maximum	customer	external_entity_paradise	<input type="checkbox"/>

This page provides you the names of rulesets, description, aggregation, source entity, and target entity details. For more information, see [Table 18](#).

- To enable the Rulesets, click the check box against the required ruleset.
- To remove the rulesets from the list, click Delete  against the required ruleset.
- To modify the rulesets, click the Ruleset name on the list. The Ruleset Details page is displayed.
- Modify the required information and click Save. The Ruleset details are updated. For more information on the fields, see [Table 18](#).

8.2.2 Create Rulesets

Each ruleset comprises of multiple rules. The ruleset compares the attributes that are defined in the rules for the source entity with the target entity. For example, Customer to Customer, Customer to Derived Entity, Derived Entity to Derived Entity, and so on.

Rulesets are created using source entity and target entity such as customer, account, and so on. Every entity has attributes such as email IDs, the date on birth, jurisdiction, and so on. To derive a match and create a similarity edge on the graph, you must apply the conditions for these attributes such as match type, scoring method, threshold score, and weightage. When you execute the ETL, a match is created based on the new ruleset.

FCC Studio offers ready-to-use rulesets, However, a new ruleset is created based on the business requirement to get a match between new entities, to apply new scoring methods, apply new threshold score, and so on.

To create a new ruleset, follow these steps:

- On the Ruleset page, click Create. The Ruleset Details page is displayed.

- Enter the ruleset fields information as tabulated [Table 18](#).

Field	Description
Name	<p>Enter the name of the ruleset.</p> <p>NOTE:</p> <ul style="list-style-type: none"> • Enter up to 30 characters. • A ruleset cannot have the same name.
Description	<p>Enter the description of the ruleset. For example, Match Customer to Customer is based on its attributes.</p> <p>NOTE: Enter up to 60 characters.</p>
Scoring Aggregation Type	<p>Select the scoring aggregation method. By default, the maximum scoring aggregation method is selected. This scoring aggregation considers the highest score obtained out of all the rules created for a ruleset.</p>
Threshold	<p>Enter the threshold value for a ruleset. This is cumulative of all the attributes values entered in the rule for entity match. 1 stands for 100%.</p> <p>Auto Threshold: Enter values from 0 to ≤ 1. Enter a matching percentage above which you want the matches to get into the graph automatically.</p> <p>Manual Threshold: Enter values from 0 to ≤ 1. Enter a value above which you want to do borderline decisions of similarity edges. It is considered as a range greater than the Manual threshold and less than the Automatic threshold.</p> <p>The threshold scoring logic works in the following way:</p> <ul style="list-style-type: none"> • A total threshold score of less than 0.7 (70%) must be discarded. • A total threshold score of more than or equal to 0.8 (80%) must create a similarity match automatically. This is configured using the Automatic Threshold in Ruleset. • A total threshold score between 0.7 (70%) and less than 0.8 (80%) must be decided manually by the user. This is configured using Manual Threshold in Ruleset. <p>Users can increase the threshold values based on false positive matches. If there are more false positives, it means the threshold value is less. To get more positive matches, increase the threshold value.</p> <p>A Similarity Edge is generated only when the maximum score obtained for a ruleset is equal to or higher than the auto threshold value.</p> <p>NOTE: The Manual Threshold value must be less than the Auto Threshold.</p>
Source	<p>Select the source entity (node). For example, customer, account, address, and so on.</p> <p>NOTE: The values are auto-populated from the metadata table that contains the Elasticsearch, index names generated as a result of running the Sqoop job. For more information, see Sqoop Job.</p>
Target	<p>Select the target entity (node). For example, customer, account, address, and so on.</p> <p>NOTE: The values are auto-populated from the metadata table that contains the elastic search index names generated as a result of running the Sqoop job. For more information, see Sqoop Job.</p>

3. Click Create. The Rules Details page is displayed to create the rules for the new ruleset. For more information, see [Create Rules in a Ruleset](#).


8.2.3 Create Rules in a Ruleset

Every Ruleset has a set of rules such as rule threshold, scoring method (default or Jaro Winkler), match type (exact or fuzzy), and source and target attribute (Country, Email, Date of birth). These attributes help to create a set of rules for a ruleset.

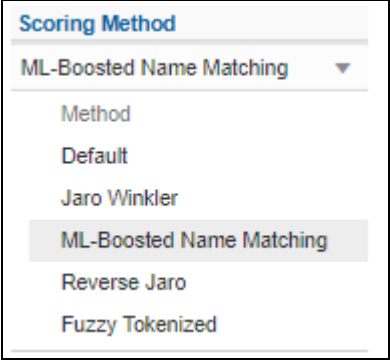
To create rules in a ruleset, follow these steps:

1. On the Ruleset Details page, click Create. A New Rule section is displayed.
2. Enter rules information as tabulated in [Table 19](#).

Field	Description
Name	Enter the name of the rule. NOTE: The maximum value must not exceed 60 characters.
Description	Enter the description of the rule. NOTE: The maximum value must not exceed 60 characters.
Rule Threshold	Enter the threshold value for a rule. The cumulative threshold value of all attributes in this rule must not exceed this value. NOTE: The values must be from 0 to <=1. This rule contributes to the matching only when the maximum score obtained for a rule is equal to or higher than the threshold value.

1. Click Save . A new rule is created.
2. To map a rule to the attributes (date of birth, email ID, jurisdiction, and so on), click Create. You can map one or more attributes to the rule.
3. Enter attributes information as tabulated in [Table 20](#).

Field	Description
Source Attribute	Select the source attribute. For example, date of birth, email ID, entity type, jurisdiction, tax ID, and so on. NOTE: Attributes are displayed based on the source entity selected when creating a rule. For example, address, event, and so on.
Target Attribute	Select the target attribute. For example, entity type, jurisdiction, tax ID, and so on. NOTE: Attributes are displayed based on the source entity selected when creating a rule. For example, customer, account, and so on.

Match Type	<p>Select one of the following match types:</p> <ul style="list-style-type: none"> • Exact: To obtain the matches that are 100% perfect when finding the entities in a database. • Fuzzy: To obtain the matches that are less than or equal to 100% perfect when finding the entities in a database.
Days	This field is deprecated.
Scoring Method	<p>Select the scoring methods:</p> <ul style="list-style-type: none"> • Default: Uses elasticsearch algorithms. • Jaro Winkler: A measure of similarity between two strings. • ML-Boosted Name Matching: Used matching learning model to identify similarity between 2 strings. • Reverse Jaro: Identify similarity between strings using a tokenized approach. • Fuzzy Tokenized: Identify similarity between tokens of a strings in a matrix based approach.  <p>For more information, see the Scoring Method.</p> <p>NOTE: ML Boosed Name Matching is only applicable to identify name matching in the 80820 Release.</p>
Threshold	<p>Enter the threshold score. It indicates the score value provided to the attribute. The sum of the total threshold value of all attributes must be from 0 to <=1. It indicates that a score below the mentioned value does not generate a result from the elasticsearch.</p>
Weightage	<p>Enter the weightage. It indicates the weightage given for the attributes in the rule.</p> <p>NOTE: Cumulative of attributes' weightage must not exceed 1.</p>
Condition	<p>Select the condition Must. It indicates that this attribute cannot have a null value. This attribute must be populated and must return a value for the matching.</p>

NOTE

- Source attribute, target attribute, and weight cannot be null.

- Rules cannot have the same name.

4. Click Save. The new ruleset is created.

8.2.4 Scoring Method

The scoring methods used in the entity resolution component are as follows:

- Default Method: The distance is computed by finding the number of edits that transform one string to another. The transformations allowed are as follows:
 - Insertion: Adding a new character
 - Deletion: Deleting a character
 - Substitution: Replace one character with another

By performing these operations, the algorithm attempts to modify the first string to match the second one. The final result obtained is the edit distance.

For example:

```
textdistance.levenshtein('arrow', 'arow')
1
>> textdistance.levenshtein.normalized_similarity('arrow',
'arow')
0.8
```

If you insert a single 'r' in string 2, that is, 'arow', it becomes the same as the string 1. Hence, the edit distance is 1. Similar to Hamming distance, you can generate a bounded similarity score between 0 and 1. The similarity score obtained is 80%.

- Jaro Winkler: This algorithm gives high scores for the following strings:
 - The strings that contain the same characters, but within a certain distance from one another.
 - The order of the matching characters is the same.

The distance of finding a similar character is one character less than half of the length of the longest string. So, if the longest string has a length of five, a character at the start of the string 1 must be found before or on $((5/2)-1) \sim 2$ nd position in the string 2. This is considered a valid match. Hence, the algorithm is directional and gives a high score if matching is from the beginning of the strings.

For example:

```
textdistance.jaro_winkler("mes", "messi")
0.86
b. textdistance.jaro_winkler("crate", "crat")
0.96
```

```
c. textdistance.jaro_winkler("crate", "atcr")
0.0
```

- First case (a): As the strings are matching from the beginning, a high score is given.
- Second case (b): Only one character was missing and that too at the end of the string 2, hence a very high score is given.
- Third case (c): The last two characters of string 2 are rearranged by bringing them at the front and hence results in 0% similarity.
- ML Boosted Name Matching: This scoring method uses Machine Learning to calculate a similarity score between two candidate name strings.
 - It uses the Catboost model, which is an ensemble of decision trees built using gradient boosting embedding algorithm. The input to the model are numerical features. The features are calculated as various similarity metrics between the two candidate names.

For example, Q-gram distance, longest common subsequence, Levenshtein distance, Jaro-Winkler distance, set letter distance, Editex, and so on. The output of the model is a similarity score between 0 and 1. For this, a pre-trained model is provided. The pre-trained model is built from names from publicly available datasets depending on whether we mention the names of the datasets or not, we should use the term “publicly available” accordingly. The names are transformed, by inserting typos, concatenations, abbreviations, and so on.

The original and transformed names are matched by a smart filter to produce a list of possible matches called candidates, along with information indicating whether two names started from the same name (ground truth). Then you can calculate the numerical similarity feature metrics of the candidate names.

The name strings are dropped, and the numerical features along with ground truth is used for training of the ensemble of decision trees model (Catboost model).

- Following are the examples using the pre-trained model:
 - Name 1: Carolyn Joyce
 - Name 2: Caroline Joyce
 - Resulting score: 0.9821176658138586
- Other example,
 - Name 1: Donatella Collombini
 - Name 2: Donatello di Betto Bardi
 - Resulting score: 0.0000005237652440222384
- Reverse Jaro: This algorithm tokenises source and target tokens, then uses Jaro Winkler algorithm to calculate the score between tokens, and then consolidate the scores to a single score.
 - **Tokenization:**

- The source and target strings are tokenised with each space. For instance, if the source name is **ANTONY EDWARD STARK** and target name is **TONY STARK**, we tokenise source name to source array as **ANTONY, EDWARD, STARK** and target string as **TONY, STARK**. Financial Crime Graph Model Matching Guide | 3 Scoring Method Fuzzy Tokenized

- Scoring:

- The array is represented as a 2-d matrix, with source token array as rows and target token array as columns and then scores between those tokens will be calculated using Jaro Winkler algorithm.

Customer Details	TONY	STARK
ANTONY	94	14
EDWARD	43	32
STARK	18	100

The Maximum weighted row scores and maximum column scores will be calculated
For the example in the table,

MaximumWeightedRowScore are as follows:

- 94 (maximum of 94, 14)
- 43 (maximum of 43, 32)
- 100 (maximum of 18,100)

MaximumWeightedColScore are as follows

- 94 (maximum of 94, 43, 18)
- 100 (maximum of 14,32,100)

To get the final score, you can calculate the sum of maximumWeightedRowScore and MaximumWeightedColumnScore (**237+194**) divided by the sum of row size and column size (**3+2**)

In this case $431/5 = 86$.

- Fuzzy Tokenised: This algorithm tokenises source and target tokens, then uses Jaro Winkler algorithm to calculate the score between tokens, and then consolidate the scores to a single score.

Tokenisation:

The source and target strings are tokenised with each space. For instance, if the source name is ANTONY EDWARD STARK and target name is TONY STARK, we tokenise source name to source array as ANTONY, EDWARD, STARK and target string as TONY, STARK.

Always the string with maximum number of tokens will be passed as source token array and the other will be target token array.

Scoring:

The array is represented as a 2-d matrix, with source token array as rows and target token array as columns and then scores between those tokens will be calculated using Jaro Winkler algorithm.

The Maximum weighted row scores and maximum column scores will be calculated.

Customer Details	TONY	STARK
ANTONY	94	14
EDWARD	43	32
STARK	18	100

- STARK in third row matching against STARK in second column
- STARK in second column matching against STARK in third row

In this scenario both returns the same score for same source and target token, so we won't consider this value twice.

- Such instances will be counted as a resCount (residue Count).
- Similar to reverse jaro, MaximumWeightedColScore score is calculated, and Weighted score of Row without duplicates will be calculated as MaximumWeightedResRowScore

Then finally similar to Reverse Jaro, we calculate

$$\text{Fuzzy tokenised score} = (\text{MaximumWeightedColScore} + \text{MaximumWeightedResRowScore}) / (\text{colSize} + \text{resCount})$$

only if the score is greater than the threshold.

In other case,

Word Match Percentage is calculated. Ref

(<https://docs.oracle.com/en/middleware/fusion-middleware/enterprise-data-quality/12.2.1.4/edqoh/comparison-word-match-percentage.html>)

Then resScore is calculated,

$$\text{which is } (\text{MaximumWeightedRowScore} * (1 - \text{threshold})) / (\text{resCount} + 1)$$

After the calculations, we will count the no. of tokens in column which has score > threshold.

if (count >= 2) then the

$$\text{Fuzzy tokenised score} = \text{threshold} + (\text{word match percentage} * \text{resScore})$$

8.3 Configure a Data Source

The data source configuration allows you to view the newly added edges or nodes in the graph. Define the source of the data, specify the order in which the files must be read, and so on.

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

1. Navigate to the `fcc_studio_etl_queries` table in the Studio Schema. The FCDM related nodes and edges are available in the table.
2. If you want to add additional nodes or edges, you can add a new entry in the `fcc_studio_etl_queries` table.
3. Enter the following details in the `fcc_studio_etl_queries` table to add a new node or edge as tabulated in [Table 21](#).

Column Name	Description	Applicable For
Type	Enter the column name. Enter the value as NODE or EDGE.	Applicable for node and edge queries.
DF_NAME	Enter the name for the node or edge.	Applicable for node and edge queries.
SOURCE	Enter the source of the data. For example, FCDM or ICIJ	Applicable for node and edge queries.
DATAFRAME	Enter the properties of the node or edge. NOTE: Enter this value only if the data source is Hive and not a .csv file.	Applicable for node and edge queries.
QUERY	<p>If the source is Hive, provide the Hive query.</p> <p>If the source is a .csv file, provide the query in the following format:</p> <pre>spark.read.format("csv").option("header", "true") .option("mode", "DROPMALFORMED").load("#{FILE-PATH}").select("node_1", "node_2", "rel_type", "SourceID") .withColumn("Label", lit("address of")).withColumnRenamed("node_1", "from").withColumnRenamed("node_2", "to").withColumnRenamed("rel_type", "E</pre>	Applicable for node and edge queries.

	<pre>DGE_ - TYPE") .withColumnRenamed("S ourceID", "Source") .filter(col("EDGE_ - TYPE")=== "registered_ adre ss").with- Column("node_ID", concat(lit ("#NUMBER#"), col("node_ID")))</pre> <p>For more information, see Configure Spark Query Parameters.</p> <p>NOTE: Ensure that the source .csv file is UTF-8 compatible.</p>	
KEY_COLUMN_NAME	<p>Set the value to the column name of your unique identifier if the query is for node.</p> <p>For example: 'node_id'.</p>	Applicable for node query.
SOURCE_NODE	<p>Enter the DF_NAME of the node from which the edge starts from.</p>	Applicable for edge query.
DESTINATION_NODE	<p>Enter the DF_NAME of the node from which the edge ends.</p>	Applicable for edge query.
SOURCE_KEY_COLUMN_NAME	<p>Set the value to the column name which has key_column values of the Source Node.</p> <p>For example: 'from_id'</p>	Applicable for edge query.
DESTINATION_KEY_COLUMN_NAME	<p>Set the value to the column name which has key_column values of the Destination Node.</p> <p>For example: 'to_id'</p>	Applicable for edge query.
ACTIVE	<p>Enter the value. The expected values are 'Y' or 'N'. Set the value to Y to consider ETL and Graph loading.</p>	Applicable for node and edge queries.

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

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

4. Enter the following details in the `fcc_studio_etl_files` table to add the file path as tabulated in [Table 22](#).

Column Name	Description
DF_NAME	Enter the name of the node or edge.

DF_SEQ_NO	Enter the unique sequence ID for each file. For example, column number.
FILEPATH	Enter the path where the .csv files are stored. NOTE: If one data frame has multiple .csv files, then make separate entries for all the files. For example: see Figure: fcc_studio_etl_files Table .
FILEORDER	If data must be imported from multiple files, specify the order in which the files must be read. For example, if the query for an entity uses multiple files, then the sequence must be provided so that the file path is replaced with the path of the correct file.

The following image provides an example of fcc_studio_etl_files.

Figure :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

8.3.1 Configure Spark Query Parameters

This section provides information on the Spark query parameters that are used during configuring a new data source for a graph or modifying existing data queries for ICIJ. This can be used for modifying the spark query of ICIJ. For example, adding new attributes or adding a new data source.

NOTE This activity is optional for the ETL process while adding or modifying data sources.

To configure Spark Query Parameter, follow these steps:

1. Navigate to the `fcc_studio_etl_queries` table in the Studio Schema. The FCDM related nodes and edges are available in the table.
2. Enter the following details in the `fcc_studio_etl_files` table to add the file path as tabulated in [Table 23](#).

Query Parameter	Description
<code>spark.read.format("csv")</code>	Enter the input file format. For example: .csv.
<code>option("header", "true")</code>	Enter the presence of a header in the input file. <ul style="list-style-type: none"> • True indicates that the header is available in the input file.

	<ul style="list-style-type: none"> False indicates that the header is absent in the input file.
<code>load("Path").</code>	<ul style="list-style-type: none"> Load indicates to load the data from the mentioned file path. The path indicates the path where the files are placed. <p>You can load to multiple paths using the following format: ("Path1", "Path2", ...)</p>
<code>select("Col1", "Col2", "Col3", "Col4")</code>	Select the columns in the input file.
<code>withColumn("A", lit("Test1"))</code>	Add a new column with column name A and column value Test1.
<code>withColumnRenamed("A", "B")</code>	Rename a column with a different name. For example, rename column from A to B.
<code>filter(col("A")=="Test1")</code>	Enter the "Where" filter condition. Here, the value for column A is Test1.
<code>withColumn("B", concat(lit("Test1"), col("A")))</code>	<p>Add a new column B, whose value is the concatenated value of Test1 and column A.</p> <p>For example, Test1=ABC</p> <ul style="list-style-type: none"> Column A contains Country and Pin code as the column values. Column B gets ABC Country and ABC Pincode as column values.

8.4 Configure Graph

The FCC Studio provides an intuitive way for creating graphs used in notebooks, where you can load graphs from external sources or create custom graphs. Using PGX, you can load multiple graphs into a notebook and create PGQL queries against different graphs. The result obtained from running a paragraph in a notebook can be used as an input to other paragraphs in the notebook. The results of analytics algorithms are stored as transient properties of nodes and edges in the graph. Pattern matching can then be used against these properties.

The graph configuration can be defined through UI based configurator or a JSON configurator. Graph configurations give you easy access to graphs using PGX-ALGORITHM, PGX-JAVA, and PGQL interpreters.

Use this section to configure attributes, extra empty nodes and edges providers, and local date format for graphs.

Topics:

- [Attributes Case in Graph](#)
- [Extra Empty Nodes and Edges Providers](#)
- [Additional Configuration](#)

8.4.1 Configure Attributes in Graph

Use this section to configure the attributes of nodes and edges in the graph.

NOTE In FCC Studio, the heterogeneous graph does not support the dynamic addition of Nodes and Edges Provider in the graph. If extra nodes or edge providers are required, then you must add the entries to the `FCC_GRAPH_EMPTY_ENTITY_MAPPING` table.

All the attributes of nodes or edges must be present in `FCC_GRAPH_COLUMN_NAME_MAPPING` table.

- `COLUMN_NAME`: Indicates the attributes name in queries.
- `RENAMED_COLUMN_NAME`: Indicates the required attribute name.
- `COLUMN_DATA_TYPE`: Indicates the PGX's data type of the attribute.

NOTE

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

For example, if the values are as follows:

- `COLUMN_NAME`: `sample_attribute`
- `RENAMED_COLUMN_NAME`: `Sample_AttributeName`
- `COLUMN_DATA_TYPE`: `string`

Then the attribute name shown in the graph is, Sample_AttributeName.

The following image provides you an example.

Figure: FCC_GRAPH_COLUMN_NAME_MAPPING Table

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

8.4.2 Configure Extra Empty Nodes and Edges Providers

In FCC Studio, the heterogeneous graph does not support the dynamic addition of Nodes and Edges Provider in the graph. If extra nodes or edge providers are required, then you must add the entries to the `FCC_GRAPH_EMPTY_ENTITY_MAPPING` table.

Where,

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

For example: `column1:string,column2:long`

NOTE

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

- Example 1:
 - TYPE: NODE

- NAME: extra_node
- COLUMN_MAPPING: name:string,phone_number:integer

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

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

Extra edges are formed between every node provider including itself with the name as "`<source_node_provider>_extra_edge_<destination_node_provider>`", with the attributes, Name, Risk and Edge Type, datatype, string, long, and string respectively.

To configure extra empty nodes and edges providers:

1. Navigate to Studio schema and go to the `FCC_GRAPH_EMPTY_ENTITY_MAPPING` table.
2. Add the entries to the `FCC_GRAPH_EMPTY_ENTITY_MAPPING` table.

The following image provides you an example.

Figure: FCC_GRAPH_EMPTY_ENTITY_MAPPING Table

NAME	TYPE	COLUMN_MAPPING
1 searched_entity	NODE	source:string,label:string,name:string,address:string,tax_id:string,date:string,initialShowPropName:string
2 is_similar_to	EDGE	label:string,match_weight:float,match_score:string

8.4.3 Additional Configuration (Local Date Format)

Use this section to configure the local date format.

To configure the local date format, follow these steps:

1. Navigate to Studio schema and go to the `FCC_DATASTUDIO_CONFIG` table.
2. For the ready-to-use graph's configuration, the following parameters are set in the `FCC_DATASTUDIO_CONFIG` table:
 - a. `local_date_format`: The default value: [M/D/YYYY, M-D-YYYY, D/M/YYYY, D-M-YYYY, YYYY-MM-DD, YYYY/MM/DD, YYYY-D-M, or YYYY/D/M].

NOTE The date format option can be used only to view the data type of an attribute on the graph in the configured format.

- b. `vertex_id_type`: The default value is "long" as per the ready-to-use queries.
This parameter represents the datatype of the `vertex_id` column or `key_column` of node providers.

NOTE This data type should be consistent across all nodes.

8.5 Apply Graph Fine-Grained Access Control

The Graph Fine-Grained Access Control and Redaction changes are applied to the FCC Studio to redact the sensitive data in the Graph and provides role-based access control, which restricts the graph access at Business domain and Jurisdiction level of the user.

DSREDACT role enables data redaction feature for any user. In case, you are using OFSAA for user creation, then map the user you want redact to be applied with DSRedact role.

To apply Graph Fine-Grained Access Control, follow these steps:

1. For OFSAA user creation, navigate to OFSAA Identity Management page and map the user to the DSREDACT role.

For SAML user creation, map the required user group to DSREDACT

2. Data redaction and fine grain access (Business Domain and Jurisdiction filters) on the Graph is configurable.
 - `fcc_studio_redaction_mapping` table is used to manage turning On and Off of Data redaction and fine grain access features.
 - If the Graph data is to be filtered based on Business domain and Jurisdiction associated with the user. Then set “Jur_BusDmn_Rule” role in the table to 'Y', else set it to 'N'
 - To enable the redaction feature on the graph, then set “DSREDACT” role in the table to 'Y' else 'N'.
3. There are ready-to-use graph properties specified for data redaction that are available in `FCC_STUDIO_REDACTION_RULE` table. To add new properties for redaction, you can specify details in the `FCC_STUDIO_REDACTION_RULE` table.

The following are the column names:

- `RULE_SEQ_ID`: Unique sequence ID
 - `LABEL`: Node or Edge label name
 - `PROPERTY`: Property that you want to redact
 - `TYPE`: Node or Edge based on the property's expected value.
4. Navigate to Studio installed server and set a variable with name `FIC_DB_HOME` as,
 5. `Export FIC_DB_HOME=<FCC_STUDIO_HOME>/ficdb`
 6. Navigate to the `<FIC_DB_HOME>/bin` directory.
 7. Run the `FCCM_Studio_ApplyGraphRedaction.sh` file. The Graph Fine-Grained Access Control changes are applied.

NOTE

Whenever you enable or disable jurisdiction filter, `FCCM_Studio_ApplyGraphRedaction.sh` has to be executed.

9 Execute ETL

Use this chapter to prepare and perform the batches to execute the ETL process. You can also verify the status of all tasks at the end of batch execution. You can verify both the overall status of the batch as well as individual task status. Execute the ETL by preparing and running the batches. You can also verify the status of all tasks at the end of batch execution. You can verify both the overall status of the batch as well as individual task status.

Topics:

- [Prepare Batches](#)
- [Perform Batches](#)
- [Verify Batch Execution](#)

9.1 Prepare the Batches

Use this section to prepare batches to execute the ETL. Batches enable you to load graphs, run notebooks, and move data from Oracle Database or Big Data to FCC Studio. Batches are prepared based on the Realms you are using.

- [Prepare Batches for FCCRealm](#)
- [Prepare Batches for FCCSAML Realm](#)

9.1.1 Prepare Batches for FCCRealm

To prepare the batches for FCCRealm, follow these steps:

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

or

- `FCCM_Studio_Set_UserPass.sh -u "USERNAME" -p "PASSWORD"`

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

NOTE

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

9.1.2 Prepare Batches for FCCSAML Realm

To prepare the batches for `FCCSamlRealm`, you must generate an API token and configure the `NBExecutor.properties`.

To generate the API token, follow these steps:

1. Navigate to the `<Studio_Installation_Path>/ficdb/bin` directory.
2. Run the shell script: `fic_db_home/FCCM_Studio_Generate_APIToken.sh`.
3. Run the script: `fic_db_home/FCCM_Studio_Generate_APIToken.sh APPNAME`.

For example, use the `./FCCM_Studio_Generate_APIToken.sh BATCH_USER`.

In the `NBExecutor.properties` file, specify the following details:

- `saml=true`
- `username=<BATCH_USERNAME>`
- `password=<BATCHUSER_PASSWORD>`
- `apiToken=<API_TOKEN>`

NOTE

`BATCH_USERNAME` and `BATCHUSER_PASSWORD` can be NULL.

9.2 Perform the Batches

Batches are performed to execute the ETL process. The batches contain Sqoop Job, Connector Job, Graph Job, and Similarity Edge Generation Job in OFSAAI. You can also execute the ETL process by running the scripts without configuring the batches.

You can perform batches in the following ways:

- **Perform batches using OFSAAI:** For more information, see [Create and Execute Run Executable](#).
- **Perform batches using shell script:** For more information, see relevant jobs in [Run ETL](#).

9.2.1 Run ETL

To run the ETL, you must perform these jobs:

- [Sqoop Job](#)
- [Connector Job](#)
- [Graph Job](#)
- [Similarity Edge Generation Job](#)

NOTE You must not trigger the same ETL job twice until it is completed.

9.2.2 Sqoop Job

Sqoop is a tool designed for efficiently transferring bulk data between Hadoop and structured datastores such as relational databases. Sqoop job creates and saves the import and export commands. It specifies parameters to identify and recall the saved job. This re-calling or re-executing is used in the incremental import, which can import the updated rows from the RDBMS table to HDFS.

- NOTE**
- This section is applicable for FCC Studio with non-OFSAA.
 - Before performing a Sqoop job, verify the Schema creation (it replicates the table structure of FCDM tables from atomic schema into HIVE schema).

The Sqoop Job moves data from BD/ECM Atomic tables to Hive tables based on the date range. This task can be skipped in the graph if FCDM data is not required.

To execute the Sqoop job, follow these steps:

1. Navigate to the `FIC_DB_HOME/bin` directory.
2. If this is your first Sqoop job, execute the following command.

```
./FCCM_Studio_SchemaCreation.sh HIVE
```
3. The Sqoop job can be scheduled or executed using the following command:

NOTE This example is applicable to shell script.

```
./FCCM_Studio_ETL_SqoopJob.sh <FROM_FIC_MIS_DATE>  
<TO_FIC_MIS_DATE> SNAPSHOT_DT<=SNAPSHOT_DATE> <Batch_ID>
```

For example:

```
./FCCM_Studio_ETL_SqoopJob.sh "20151201" "20200412"  
"SNAPSHOT_DT=20200415" "BatchID_001"
```

Where:

- FROM_FIC_MIS_DATE is 20151201
- TO_FIC_MIS_DATE is 20200412
- SNAPSHOT_DT is 20200415
- Batch_ID is BatchID_001

NOTE The date format is "YYYYMMDD"

If the date parameters are passed as null, then the values of these parameters are calculated based on ETL_PROCESSING_RANGE, and the date's value is as follows:

- Snapshot_dt is considered as the current date.
- To_fic_mis_date is considered as a yesterday's date.
- From_fic_mis_date is considered as a date which is etl_processing_range behind to_fic_mis_date.

For example:

```
./FCCM_Studio_ETL_SqoopJob.sh "null" "null"  
"SNAPSHOT_DT=null" "BatchID_001"
```

If, the ETL processing range: 2Y, 3M, 10D (2 years, 3 months, 10 days) and Present Date: 20200815, then:

- Snapshot_dt is 20200815
- To_fic_mis_date is 20200814
- From_fic_mis_date is 20180504

9.2.3 Connector Job

The connector job transforms the data from the Hive table or the .csv files based on data source into the node and edge format and recognizes the changes in data for the graph.

To execute the connector job, follow these steps:

1. Navigate to the FIC_DB_HOME/bin directory.

2. Execute the following command:

```
./FCCM_Studio_ETL_Connector.sh <Source>  
SNAPSHOT_DT=<SNAPSHOT_DATE>
```

NOTE The date format is “YYYYMMDD”

For example,

```
./FCCM_Studio_ETL_Connector.sh FCDM SNAPSHOT_DT=20200415
```

Where:

- Source: FCDM
- Snapshot DT: 20200415

FCC Studio has a ready-to-use configuration for FCDM and ICIJ data source as per the graph model. If the date parameter is passed as null, then the snapshot date is taken from the previous Sqoop job if present, otherwise it is present day.

For example:

```
./FCCM_Studio_ETL_Connector.sh FCDM SNAPSHOT_DT=null
```

The snapshot date is 20200815 (refer the example from [Sqoop job](#))

- For ready-to-use, run the following command for FCDM.

```
./FCCM_Studio_ETL_Connector.sh FCDM SNAPSHOT_DT=20200415
```

- For ready-to-use, run the following command for ICIJ.

```
./FCCM_Studio_ETL_Connector.sh ICIJ SNAPSHOT_DT=20200415.
```

NOTE When the connector snapshot date is 'Null' then it takes a snapshot of the date of the last run Sqoop job.

9.2.4 Graph Job

The Graph Job task generates the JSON files for the PGX server to load with other .csv files for all the sources and updates the changes into the PGX server.

To execute the Graph job, follow these steps:

1. Navigate to the FIC_DB_HOME/bin directory.
2. Execute the following command:

```
./FCCM_Studio_ETL_Graph.sh.
```

After the first execution of this task, start the PGX server to load the graph, which can be queried and viewed in the FCC Studio Notebook.

9.2.5 Similarity Edge Generation Job

This task generates the similarity edges based on the ruleset. The similarity edges above the manual threshold are also added to the Graph.

To execute the Similarity Edge Generation job, follow these steps:

1. Navigate to the `FIC_DB_HOME/bin` directory.
2. Execute the following command:

```
./FCCM_Studio_ETL_BulkSimilarityEdgeGeneration.sh.
```

NOTE This job must be triggered when you restart their PGX server.

9.2.6 Similarity Edge Generation Job for Infer API (ML Name Boosted Matching Method)

To trigger similarity edge job with **ML Name Boosted** scoring method, perform the following steps:

1. Navigate to the `FIC_DB_HOME/bin` directory.
2. Export the `TNS_ADMIN=<wallet location>`
3. Export the `LD_LIBRARY_PATH=<Oracle instance client path>`

For example, export

```
LD_LIBRARY_PATH=/opt/oracle/instantclient_19_8:$LD_LIBRARY_PATH
```

NOTE You must ensure that you have selected ML Name Boosted Matching as a Scoring Method in the Ruleset Configuration. For more information, see [Scoring Method](#).

4. Execute the following command:

```
./FCCM_Studio_ETL_BulkSimilarityEdgeGeneration.sh.
```

NOTE You must do these export operations before starting the Studio service.
These steps can be performed only for the **On-Premises Studio**.

9.2.7 Similarity Edge Generation Job for Training API

To trigger the Training API, perform the following steps:

1. Navigate to the `FIC_DB_HOME/bin` directory.
2. Export the `TNS_ADMIN=<wallet location>`
3. Export the `LD_LIBRARY_PATH=<Oracle instance client path>`

For example, export
LD_LIBRARY_PATH=/opt/oracle/instantclient_19_8:\$LD_LIBRARY_PATH

NOTE You must do these export operations before starting the Studio service.
These steps can be performed only for the **On-Premises Studio**.

4. Execute the following command:

```
./FCCM_Studio_ML_Model_Training.sh name 0.6
```

So, 'name' is modelType Here and '0.6' is accuracyThresold for Training Model.

Once you triggered this to train model there can be following scenario ->

- If Model Accuracy comes more than 0.6 -> model will be trained and published with new accuracy and new records will be inserted into table 'FCC_ER_MODEL_NAME_BOOSTED' and this model will become champion model(column isChampion is 'Y').
- If Model Accuracy come less than or equal to 0.6 -> model will be trained but will not publish with new accuracy and new records will be inserted into table 'FCC_ER_MODEL_NAME_BOOSTED' and old model will remain champion model.

NOTE Here accuracyThresold 0.6 is used for reference only. You can use any threshold.

5. To use the updated model, you must trigger, by executing the following command:

```
./FCCM_Studio_ETL_BulkSimilarityEdgeGeneration.sh.
```

9.3 Verify Batch Execution

Use this section to verify the status of all tasks at the end of batch execution. You can verify both the overall status of the batch and individual task status.

Topics:

- [Verify Sqoop Job](#)
- [Verify Connector Job](#)
- [Verify Graph Job](#)
- [Verify Similarity Edge Generation Job](#)
- [Verify Oracle Schema Tables](#)

9.3.1 Verify Sqoop Job

Use this section to verify logs and Hive tables' sqoop job.

9.3.1.1 Verify Logs

To verify logs in Sqoop Job, follow these steps:

1. Navigate to the `<Studio Home>/logs/` directory.
2. Open the `batchservice.log` file. The overall status and individual status of each moved table are displayed. Also, errors are displayed if any individual table has failed.

Based on this you can fix accordingly or contact [My Oracle Support](#) in case of any errors.

9.3.1.2 Verify Hive Tables

To verify the Hive table in Sqoop Job, follow these steps:

1. Connect to the Hive Schema.
2. Verify if data was moved into the respective tables (based on logs) for the snapshot date of batch.

9.3.2 Verify Connector Job

Use this section to verify logs and Hive tables for the connector job.

9.3.2.1 Verify Logs

To verify logs in Connector Job, follow these steps:

1. Navigate to `<Studio Home>/logs/`
2. Open the `batchservice.log` file. The overall status and status of each entity is displayed. Also, errors are displayed if any entity has failed.

Based on this you can fix accordingly or contact [My Oracle Support](#) in case of any errors.

9.3.2.2 Verify Hive Tables

To verify the Hive table in Connector Job, follow these steps:

1. Connect to the Hive Schema
2. Verify if table names: `<Source>_<entity_name>` (example: `fcdm_customer`) are present and populated or not based on the log.

9.3.2.3 Verifying Indices in Elasticsearch

To verify indices in the elasticsearch, follow these steps:

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

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

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

2. Format: `<Index name>_<Snapshot Date>`

For example:

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

9.3.3 Verify Graph Job

Use this section to verify logs and Hive tables for graph job.

9.3.3.1 Verify Logs

To verify logs in the Graph Job, follow these steps:

1. Navigate to the `<Studio Home>/logs/ >` directory.
2. Open the `batchservice.log` file. The overall status and status of each entity is displayed. Also, errors are displayed if any entity has failed.

Based on this you can fix accordingly or contact [My Oracle Support](#) in case of any errors.

For example, if the fix requires a query change or configuration changes, follow the cleanup steps and re-run the tasks after fixing it.

9.3.4 Verify Similarity Edge Generation Job

Use this section to verify logs and Hive tables for the Similarity Edge Generation job.

9.3.4.1 Verify Logs

To verify logs in the Graph Job, follow these steps:

1. Navigate to the `<Studio Home>/logs/ >` directory.
2. Open the `batchservice.log` and `entity-resolution.log` file. The overall status and status of each ruleset is displayed. Also, errors are displayed if any entity has failed.

Based on this you can fix accordingly or contact [My Oracle Support](#) in case of any errors.

9.3.5 Verify Oracle Schema Tables

To verify logs in the Graph Job, follow these steps:

1. Navigate to the Oracle Studio schema.
2. Verify if the similarity edges are formed for the following tables:
 - `fcc_er_matched_edges`
 - `fcc_er_matched_edges_manual`

9.3.6 Clean up for ETL

If any ETL jobs are failed, and you want to re-run the job you must clean up the ETL.

To clean up the ETL, follow these steps:

1. Navigate to FCC Studio schema based on the source (for example, FCDM, ICIJ), and delete the following tables.
 - fcc_studio_graph_entity_provider
 - fcc_studio_graph_plug_edge_status
2. Drop the tables created in Hive schema.
 - If you want to clean up the ICIJ job, then drop ICIJ related tables. For example, icij_paradise_external_entity.
 - If you want to clean up the FCDM job, then drop FCDM related tables. For example, fcdm_customer.
3. Truncate the tables created after the schema creation job (For example, cust, acct, wire_trxn, fcc_studio_nodeedge_lookup, and so on)
4. Delete the folder where graph.json and .csv files are created that is, "HDFS_GRAPH_FILES_PATH" (refer config.sh under the <Studio installation path>/bin path.

10 Monitor Tasks

Tasks are created when notebooks or paragraphs are executed by the Notebook users. It is important to know the status of the execution whether the tasks are created, rejected, cancelled, and so on. The Tasks page allows you to view the status of the task and associated notebooks, paragraphs, interpreters, and so on. By default, all the tasks are listed on the Task page. You can view the specific task using filters such as status of the task, date of creation, name of the notebook.

Topics:

- [View Tasks Using Status](#)
- [View Tasks Using Time of Creation](#)
- [View Tasks Using Names of Notebook](#)

10.1 View Tasks Using Status

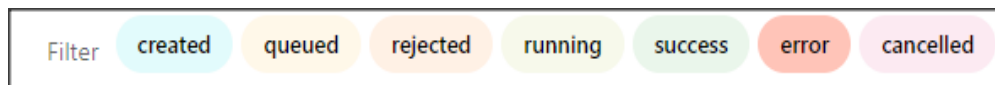
Use this section to filter the tasks by their status, for example, created, rejected, queued, and canceled, and so on.

To view task using the status filter, follow these steps:

1. In the Crime & Compliance Studio menu list, click Tasks. By default, the Tasks page lists all the tasks of FCC Studio and displays the notebook, paragraph, interpreter, and user associated with each task. The status filters are displayed on the top of the page. For example, Created, Rejected, and so on.

Notebook	Paragraph	Interpreter	Status	Creation Time	Queue Time	Run Time	User
TestData	SELECT n.Label, count(n.Label) FROM ...	pgql	success	11 Aug 2020 01...	218ms	5s 474ms	FCCMDSUSE
Financial Crime Graph Patterns	Loading the GlobalGraph	pgx-java	success	11 Aug 2020 01...	110ms	2s 73ms	FCCMDSADI
TestData	SELECT e.Label FROM GlobalGraphH ...	pgql	success	11 Aug 2020 12...	28ms	2s 74ms	FCCMDSUSE
TestData	SELECT n.Name, n.Label FROM Global...	pgql	success	11 Aug 2020 12...	25ms	1s 569ms	FCCMDSUSE
TestData	SELECT e.Label, count(e.Label) FROM ...	pgql	success	11 Aug 2020 12...	37ms	1s 471ms	FCCMDSUSE
TestData	SELECT n.Label, count(n.Label) FROM ...	pgql	success	11 Aug 2020 12...	299ms	9s 788ms	FCCMDSUSE

2. To filter the tasks based on the status, click the required status or statuses (created, error, rejected, and so on).



The tasks in those statuses are displayed with the associated Notebooks, Paragraphs, and Interpreters.

[Table 24](#) describes the list of statuses.

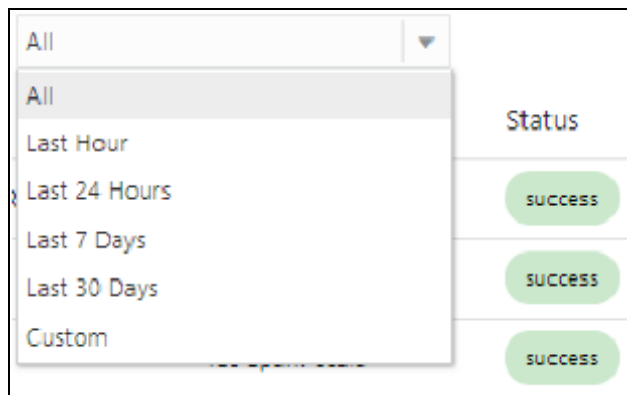
Field	Description
created	The task is just created.
queued	The task is in a queue, waiting to be run. This can happen when the same user runs multiple paragraphs of the same interpreter in the same notebook - the interpreter will first finish executing the first paragraph (that is., task) and then moves to the second task which will have status queued until then.
running	The tasks are being executed.
rejected	The task is rejected.
success	The task is completed.
canceled	The execution of the task is canceled (for example, by clicking the 'Cancel Execution' ("Stop") button on a paragraph).
error	An error occurred during the execution of a task. The error can be one of the following: <ul style="list-style-type: none"> • The concerned interpreter is unsupported • The interpreterClient is disconnected • The task is not found • The status of the task cannot be changed to running or success

10.2 View Tasks Using Time of Creation

Use this section to filter the tasks by their time of creation, for example, last hour, last 24 hours, last 7 days, and so on.

To filter tasks using date and time, follow these steps:

1. Select the required task creation time or days, for example, last hour, last 24 hours, last 30 days, and so on.




The following is the list of available options

- Last Hour
 - Last 24 Hours
 - Last 7 days
 - Last 30 days
 - Custom: Allows the user to enter a specific date range (From and To date-time). If it is empty, it assumes an infinite past or future.
2. To specify the date range, click Custom. The date range is displayed.

Custom

▼

From

To
 3. Select date range and time using the calendar . The respective tasks are displayed.

10.2.1 View Tasks Using Names of Notebook

Use this section to filter the tasks by the search field. This filter allows you to filter the tasks based on the notebook name, paragraph title, paragraph code, interpreter, and user.

To filter Tasks, follow these steps:

1. Enter the name of paragraph code, notebook, interpreter, or user in the Search field.

Type to search

The respective tasks are displayed in the Task list.

[Table 25](#) describes the name of the notebook, paragraph, and interpreter.

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

User	The username of the user who created the task.
------	--

2. Click Refresh  to get the latest list.

11 Restart Services

Use this section to understand how to stop or start the service in the FCC Studio in case you have an issue with the services.

Topics:

- [Stop and Start the FCC Studio Services](#)
- [Stop and Start the PGX Service](#)

11.1 Stop and Start the FCC Studio Services

To stop the FCC Studio installer, follow these steps:

1. Navigate to the `<Studio_Installation_Path>/bin/` directory.
2. Run the following command:

```
./stop-script.sh.
```

To start the FCC Studio services, follow these steps:

1. Navigate to the `<Studio_Installation_Path>/bin/` directory.
2. Execute the following command in the console:

```
./fcc-studio.sh
```

11.2 Stop and Start the PGX Service

To stop the PGX service, follow these steps:

1. Navigate to the `<PGX_Installation_Path>/pgx/server/bin` directory.
2. Run the following command:

```
Run ./stop-script.sh
```

To start the PGX service, follow these steps:

1. Navigate to the `<PGX_Installation_Path>/pgx/server/bin` directory.
2. Run the following command:

```
nohup ./start-pgx.sh &
```
3. After the PGX service runs successfully, run the

```
./FCCM_Studio_ETL_BulkSimilarityEdgeGeneration.sh job
```

 and

```
<FCCM_Studio path>/FCCM_Studio_ApplyGraphRedaction.sh
```

 file.

NOTE Ensure that Global graph is loaded in the PGX Server.

12 Configure Quantifind

Quantifind integration for real-time risk reports in FCC Studio enables the Financial institutions to discover signals of revenue drivers and risk, including fraud and money. To configure Quantifind after the installation, perform the following steps:

1. In the `Studio_Home/bin` directory, configure proxy settings in `fcc-studio.sh` file.
2. Before this `"sh "$APP_HOME"/interpreters/bin/start-all-
interpreters.sh"` command.
3. Add the below line for Http proxy configuration, replace the place holders with actual values.

```
export JAVA_OPTS="$JAVA_OPTS -
Dhttps.proxyHost=##HTTPS_PROXY_HOST## -
Dhttps.proxyPort=##HTTPS_PROXY_PORT##
-Djava.net.useSystemProxies=true"
```

In case, if proxy has any username and password then include the following command:

```
export JAVA_OPTS="$JAVA_OPTS -
Dhttps.proxyHost=##HTTPS_PROXY_HOST## -
Dhttps.proxyPort=##HTTPS_PROXY_PORT##
-Dhttp.proxyUser=##HTTP_PROXY_USERNAME## -
Dhttp.proxyPassword=##HTTP_PROXY_PASSWORD##
-Djava.net.useSystemProxies=true"
```

4. Configure Quantifind parameters in `Studio_home/interpreters/conf/quantifind.properties` directory:
 - `Quantifind-url`: `quantifind url`
 - `Quantifind_token`: `Encodedentifind_token`
 - `quantifind_appname`: `app name provided by quantifind`
 - `quantifind_enabled`: `true`
5. Similarly, configure the same fields in the `Studio_home/datastudio/server/conf/application.yml` directory.

Quantifind:

- `url`: `quantifind url`
- `token`: `Encodedentifind_token`
- `appname`: `app name provided by quantifind`
- `enabled`: `true`

6. To encode `quantifind token`. Navigate to `fic_db_home/bin` directory and execute the shell script:

```
./ FCCM_Studio_Base64Encoder.sh <Quantifind_token>
```

The result will be the encoded Quantifind token that can be configured in `quantifind.properties` and `application.yml` file.

7. Restart Studio.

13 Using FCC Studio REST API

Representational state transfer is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the internet.

13.1 Management of User Session

Using the following roles API, you can manage the user session in FCC Studio:

- [POST sessions login](#)
- [POST sessions logout](#)
- [GET sessions user](#)

13.1.1 POST sessions login FCCMRealm

Makes a post request `https://<Studio URL>/20200615/sessions/login` to supply the credentials to login FCC Studio using the configured realm.

Note: This session API can be used only with **FCCMRealm**.

Request:

- Method: **POST**
- URL: `https://<Studio URL>/20200615/sessions/login`
- **Header Details:**
 - Content-Type: **application/json**
- Object: **AuthenticationDetails (Information to authenticate request)**
- Body: Displayed as shown:

```
{
  "credentials": "admin",
  "principal": "admin@oracle.com"
}
```

Responses:

- Response Code: **200**
- Status: **Resources created**
- Body: Displayed as shown:

```
{
  "authToken": "string",
  "permissions": [
    "string"
  ]
}
```

```
],
  "username": "string"
}
```

- Response Code: **401**
- Status: **The required information to complete authentication was not provided or was incorrect.**
- Body: Displayed as shown

```
{
  "code": "string",
  "message": "string"
}
```

The auth token generated can be used to perform API calls.

13.1.2 POST sessions login for SAMLRealm

Makes a post request `https://<Studio URL>/20200615/sessions/login` to supply the credentials to login FCC Studio using the configured realm.

Note: This session API can be used only with **FCCMRealm**.

Request:

- Method: **POST**
- URL: `https://<Studio URL>/20200615/sessions/login`
- Object: **AuthenticationDetails (Information to authenticate request)**
- **Header Details:**
 - Content-Type: **application/json**
 - x-api-token: **<API_TOKEN>**
- Content-type: **application/json**
- Body: Displayed as shown:

```
{
  "credentials": "API_USER",
  "principal": "roles:DSADMIN,DSUSER|jurisdiction:|bd:"
}
```

NOTE

- API token can be generated by running the `FCCM_Studio_Generate_APIToken.sh` shell script. For more information, see [Prepare Batches for FCCSAML Realm](#).

- API_User is the user that you provided in the config.sh when installing Studio.
- DSADMIN and DSUSER can be replaced with other roles based on your requirement.

Responses:

- Response Code: **200**
- Status: **Resources created**
- Body: Displayed as shown:

```
{
  "authToken": "string",
  "permissions": [
    "string"
  ],
  "username": "string"
}
```

- Response Code: **401**
- Status: **The required information to complete authentication was not provided or was incorrect.**
- Body: Displayed as shown

```
{
  "code": "string",
  "message": "string"
}
```

13.2 Management of Roles

Using the following roles API you can manage the roles in FCC Studio:

- [GET roles](#)
- [POST roles](#)
- [GET roleId](#)
- [PUT roleId](#)
- [DELETE roleId](#)

13.2.1 GET roles

Makes a get request <https://<studio URL>/20200615/roles> to get the list of roles from FCC studio.

Request:

- Method: **GET**
- **Header Details:**
 - Content-Type: **application/json**
 - x-auth-token: **<X-AUTH-TOKEN>**
- URL: **https://<studio URL>/20200615/roles**

Responses:

- Response Code: **200**
- Status: **Resource Returned**
- Body: Displayed as shown:

```
{
  "count": 0,
  "hasMore": true,
  "items": [
    {
      "entityPermissionSet": [
        {
          "entityType": "Notebook",
          "id": "dsQ1fHfW9",
          "name": "entity name",
          "permissions": [
            "view",
            "run_all",
            "update_group"
          ]
        }
      ],
      "id": "dsQ1fHfW9",
      "name": "Admins",
      "timeCreated": "2020-12-10T07:24:23.765Z"
    }
  ],
  "limit": 0,
  "offset": 0,
  "totalResults": 0
}
```

- Response Code: **400**
- Status: **InvalidParameter**
- Body: Displayed as shown


```
{
  "code": "string",
  "message": "string"
}
```

- Response Code: **404**
- Status: **Authorization failed or requested resource not found**
- Body: Displayed as shown

```
{
  "code": "string",
  "message": "string"
}
```

13.2.2 POST roles

Makes a post request `https://<studio URL>/20200615/roles` to post the list of roles to FCC studio.

Request:

- Method: **POST**
- URL: `https://<studio URL>/20200615/roles`
- **Header Details:**
 - Content-Type: **application/json**
 - x-auth-token: **<X-AUTH-TOKEN>**
- Object: **createRoleDetails**
- Body: Displayed as shown:

```
{
  "entityPermissionSet": [
    {
      "entityType": "Notebook",
      "name": "entity name",
      "permissions": [
        "view",
        "run_all",
        "update_group"
      ]
    }
  ],
  "name": "Entity"
}
```

Responses:

- Response Code: **200**
- Status: **Resources created**
- Body: Displayed as shown:

```
{
  "entityPermissionSet": [
    {
      "entityType": "Notebook",
      "id": "dsQ1fHfW9",
      "name": "entity name",
      "permissions": [
        "view",
        "run_all",
        "update_group"
      ]
    }
  ],
  "id": "dsQ1fHfW9",
  "name": "Admins",
  "timeCreated": "2020-12-10T07:38:48.635Z"
}
```

- Response Code: **400**
- Status: **InvalidParameter**
- Body: Displayed as shown

```
{
  "code": "string",
  "message": "string"
}
```

- Response Code: **404**
- Status: **Authorization failed or requested resource not found**
- Body: Displayed as shown

```
{
  "code": "string",
  "message": "string"
}
```

13.2.3 GET roleId

Makes a get request `https://<Studio URL>/20200615/roles/{roleId}` to get the list of unique request id for logging purposes to FCC studio.

Request:

- Method: **GET**
- **Header Details:**
 - Content-Type: **application/json**
 - x-auth-token: **<X-AUTH-TOKEN>**
- URL: `https://<Studio URL>/20200615/roles/{roleId}`

Responses:

- Response Code: **200**
- Status: **Resource Returned**
- Body: Displayed as shown:

```
{
  "entityPermissionSet": [
    {
      "entityType": "Notebook",
      "id": "dsQ1fHfW9",
      "name": "entity name",
      "permissions": [
        "view",
        "run_all",
        "update_group"
      ]
    }
  ],
  "id": "dsQ1fHfW9",
  "name": "Admins",
  "timeCreated": "2020-12-10T07:43:12.705Z"
}
```

- Response Code: **404**
- Status: **Authorization failed or requested resource not found**
- Body: Displayed as shown

```
{
  "code": "string",
  "message": "string"
}
```

13.2.4 PUT roleId

Makes a put request `https://<Studio URL>/20200615/roles/{roleId}` to update the roleId in FCC studio.

Request:

- Method: **PUT**
- **Header Details:**
 - Content-Type: **application/json**
 - x-auth-token: **<X-AUTH-TOKEN>**
- URL: `https://<Studio URL>/20200615/roles/{roleId}`
- Object: **UpdateRoleDetails (New data of the role that you want to update)**
- Body: Displayed as shown:

```
{
  "entityPermissionSet": [
    {
      "entityType": "Notebook",
      "name": "entity name",
      "permissions": [
        "view",
        "run_all",
        "update_group"
      ]
    }
  ]
}
```

Responses:

- Response Code: **200**
- Status: **OK**
The updates are made for the roleIds.
- Response Code: **404**
- Status: **Authorization failed or requested resource not found**
- Body: Displayed as shown

```
{
  "code": "string",
  "message": "string"
}
```

13.2.5 DELETE roleId

Makes a delete request `https://<Studio URL>/20200615/roles/{roleId}` to delete the role from FCC studio.

Request:

- Method: **DELETE**
- **Header Details:**
 - Content-Type: **application/json**
 - x-auth-token: **<X-AUTH-TOKEN>**
- URL: `https://<Studio URL>/20200615/roles/{roleId}`

Responses:

- Response Code: **204**
- Status: **No Content**
- Response Code: **404**
- Status: **Authorization failed or requested resource not found**
- Body: Displayed as shown

```
{
  "code": "string",
  "message": "string"
}
```

- Response Code: **409**
- Status: **Cannot delete role. Invalid state**
- Body: Displayed as shown

```
{
  "code": "string",
  "message": "string"
}
```

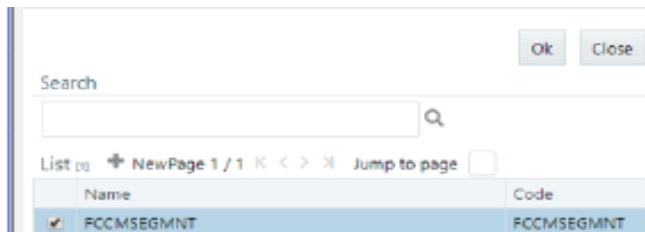
14 Appendix - Create and Execute a Run Executable

NOTE Ensure that the username and password are set before executing a notebook batch. For more information, see Prepare for Batches.

To create and execute run executable, follow these steps:

1. Log in to the OFSAA application.
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, select Rule Run Framework, and click Run. 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 Directory.

The Folder Selector dialog box is displayed.



6. Select the directory to link run executable.
7. Click OK. The Master Information toolbar is displayed.



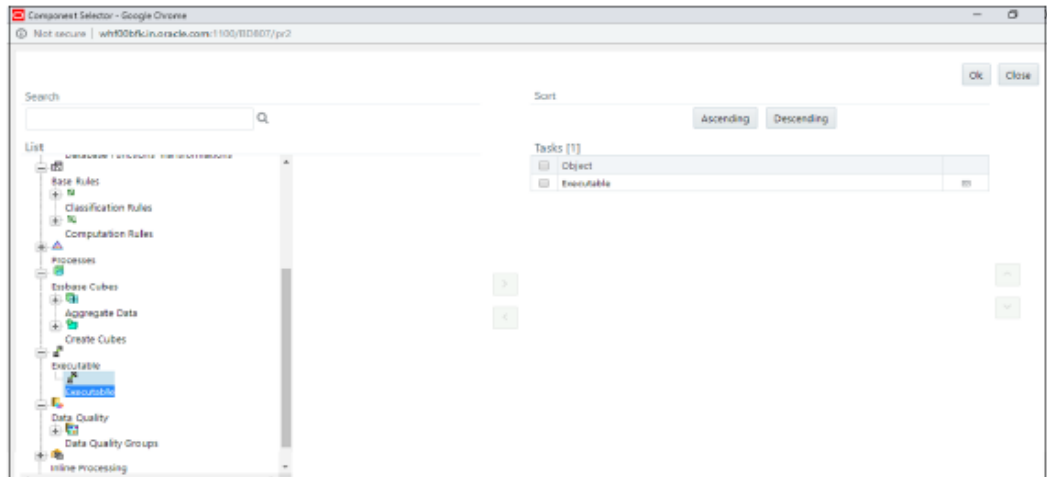
8. Enter the following details in the Master Information toolbar as tabulated in [Table 26](#).

Field	Description
Code	Enter the code of the process.

Name	Enter the name of the process.
Type	Select type for the process.

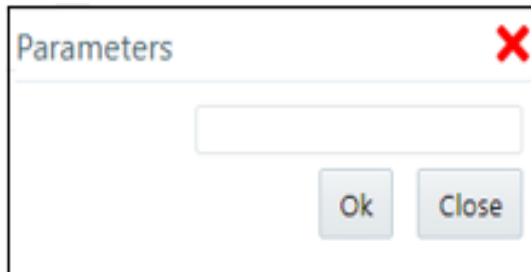
9. Click OK.
10. Click Selector in the List tree. From the options displayed, select Job. The Jobs page is displayed.
11. Click Executable on the List tree. From the options displayed, select Executable.

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 the run executable using the following commands:

Command

"FCCM_Studio_ETL_SqoopJob.sh", "
 <FROM_FIC_MIS_DATE>", "<TO_FIC_MIS_DATE>", "SNAPSHOT_DT=<SNAPSHO
 T_DATE>: Use this command for **SQOOP** job.

File Name

"FCCM_Studio_ETL_SqoopJob.sh" - Used for the **SQOOP** job.

Parameters	Description
FROM_FIC_MIS_DATE	Indicates the date from when the data movement must be performed.
SNAPSHOT_DT	Indicates the date of the data movement code and snapshot date.
TO_FIC_MIS_DATE	Indicates the date until when the data movement must be performed.

Command

"FCCM_Studio_ETL_Connector.sh", "<Source>", "SNAPSHOT_DT=<SNAPSHOT_DATE>" – Use this command for the **Connector** job.

File Name

"FCCM_Studio_ETL_Connector.sh" – Used for the **Connector** job.

Parameters	Description
Source	Indicates the data source. Ready-to-use sources are FCDM and ICIJ .
SNAPSHOT_DATE	Indicates the date of the data movement code and snapshot date.

Command

"FCCM_Studio_ETL_Graph.sh" – Used for the **Graph Loading** job.

Command

"FCCM_Studio_ETL_BulkSimilarityEdgeGeneration.sh" – Used for the **Bulk Similarity Edge Generation** job.

Command

"FCCM_Studio_NotebookExecution.sh", "notebookID", "outputParagraphID", "scenarioID", "thresholdsetID", "extraparams" – Used for batch execution of published notebook.

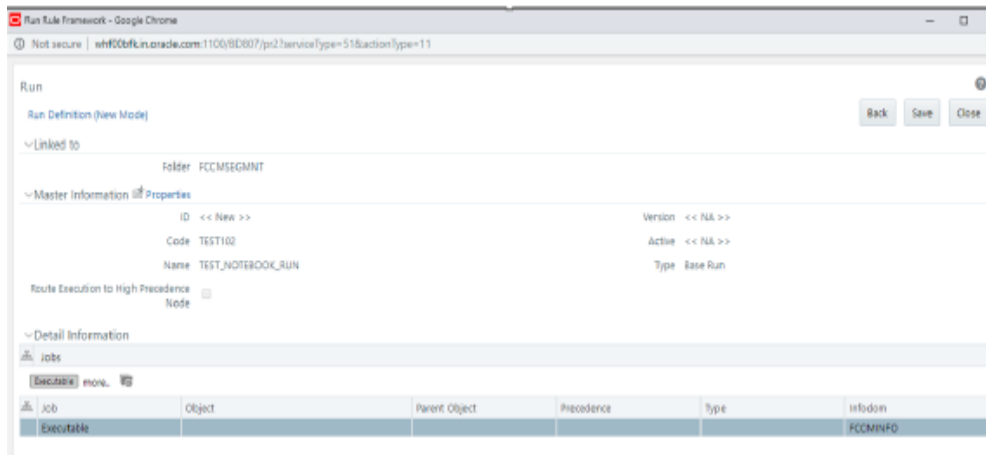
File Name

"FCCM_Studio_NotebookExecution.sh" – Used for the **Batch Execution of Published Notebook**.

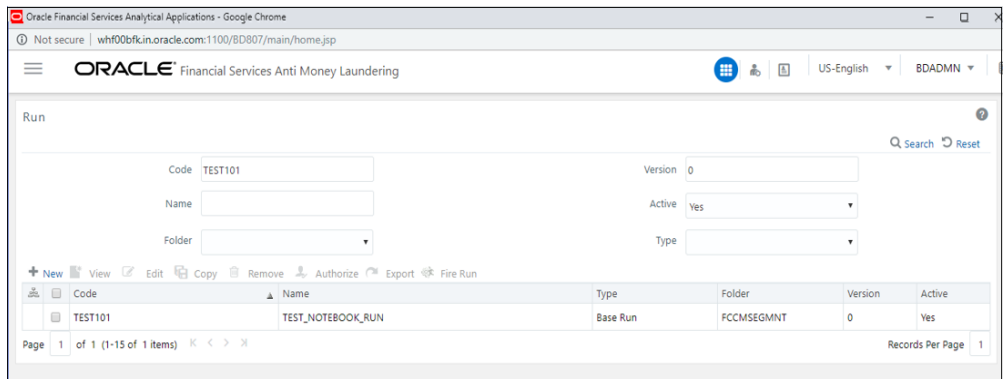
Parameters	Description
notebookId	Indicates the ID of the required notebook.

Parameters	Description
outputParagraphId	Indicates that the value is always "null"
scenarioId	Indicates the ID of Scenario
thresholdsetId	Indicates the ID of the threshold set with which notebook will run
sessionId	Indicates the ID of the session in which notebook will run
extraparams	For scenario notebook, it will be "null", but for notebook execution, it depends on the paramkeys used in the notebook

14. Click **OK**. The run executable is displayed in the **Detail Information** pane 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 Fire Run details as tabulated in [Table 28](#).

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

18. Click OK to run the Run Executable.

The Run executable starts executing.

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

The Batch Monitor window is displayed.

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

21. Click Start Monitoring in Batch Run Details.

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

15 Appendix Example of ETL

The transformed entity (node/edge) are compared and the changes are saved in form of insert and delete and then the full table is also updated so that graph can load from them in case of PGX server failure or restart.

For Example:

For the account Node: fcdm_account, the Batch 1 Transformed source data is displayed as provided in the table:

Table: Node: **fcdm_account**

node_id	label	original_id	name	risk	source	date
1000000191101	Account	AMLBMAC664	BENOY	1	FCDM	15-11-15
1000000191102	Account	AMLBMAC420	PERRY	9	FCDM	28-05-15
1000000191103	Account	AMLBMAC504	RAMESH	5	FCDM	18-12-14
1000000191104	Account	AMLBMAC654	DURKA	4	FCDM	14-12-14

On Batch 1 (first batch of ETL), since earlier data are absent, all the entries are also listed as **'insert'** and saved into hive tables with name like '<entity_provider_name>_insert', example: 'fcdm_acct_insert' and delete tables with name like '<entity_provider_name>_delete' are not created.

Table: **fcdm_account_insert**

node_id	label	original_id	name	risk	source	date
1000000191101	Account	AMLBMAC664	BENOY	1	FCDM	15-11-15
1000000191102	Account	AMLBMAC420	PERRY	9	FCDM	28-05-15

1000000191103	Account	AMLBMAC504	RAMESH	5	FCDM	18-12-14
1000000191104	Account	AMLBMAC654	DURKA	4	FCDM	14-12-14

Table: **fcdm_account**

node_id	label	original_id	name	risk	source	date
1000000191101	Account	AMLBMAC664	BENOY	1	FCDM	15-11-15
1000000191102	Account	AMLBMAC420	PERRY	9	FCDM	28-05-15
1000000191103	Account	AMLBMAC504	RAMESH	5	FCDM	18-12-14
1000000191104	Account	AMLBMAC654	DURKA	4	FCDM	14-12-14

On subsequent batch for graph, for all the nodes or edges, the comparison is made between previous batch full data and current batch full data to identify insert and delete. The update are considered as deletion of old and addition of new.

For the account Node: fcdm_account, the **Batch 2** Transformed source data is displayed as provided in the table:

Table: Node: **fcdm_account**

node_id	label	original_id	name	risk	source	date
1000000191101	Account	AMLBMAC664	BENOY	1	FCDM	15-11-15
1000000191102	Account	AMLBMAC420	PERRY	9	FCDM	28-05-15
1000000191104	Account	AMLBMAC654	DURGA	4	FCDM	14-12-14

1000000191105	Account	XXXACFRKITINGAC-009	THOMAS	7	FCDM	05-02-15
---------------	---------	---------------------	--------	---	------	----------

Here node id ending with:

- 103 has been removed
- 105 has been added
- 104 has been updated

Table: fcdm_account_insert

node_id	label	original_id	name	risk	source	date
1000000191104	Account	AMLBMAC654	DURGA	4	FCDM	14-12-14
1000000191105	Account	XXXACFRKITINGAC-009	THOMAS	7	FCDM	05-02-15

Table: fcdm_account_delete

node_id
1000000191103
1000000191104

Table: Node: fcdm_account

node_id	label	original_id	name	risk	source	date
1000000191101	Account	AMLBMAC664	BENOY	1	FCDM	15-11-15
1000000191102	Account	AMLBMAC420	PERRY	9	FCDM	28-05-15
1000000191104	Account	AMLBMAC654	DURGA	4	FCDM	14-12-14

1000000191105	Account	XXXACFRKITINGAC-009	THOMAS	7	FCDM	05-02-15
---------------	---------	---------------------	--------	---	------	----------

OFSAA Support

Raise a Service Request (SR) in [My Oracle Support \(MOS\)](#) for queries related to the OFSAA applications.

Send Us Your Comments

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, indicate the title and part number of the documentation along with the chapter/section/page number (if available) and contact the Oracle Support.

Before sending us your comments, you might like to ensure that you have the latest version of the document wherein any of your concerns have already been addressed. You can access My Oracle Support site that has all the revised/recently released documents.

