pubsubplus-connector-database

User Guide

Solace Corporation

Version 2.1.6

solace.

Table of Contents

Preface . . . 1
Management and Monitoring Micro Integration . . . 2
Configuration . . . 3
Database Source Configuration Options . . . 3
application-operator.yml . . . 3
application-db-processor.yml . . . 5
Database Sink Configuration Options . . . 7
application-operator.yml . . . 7
application-db-processor.yml . . . 9
More information . . . 11
Solace Pubsub Connector For Database Source . . . 11
1 Introduction . . . 11
2 DB Connector Configuration . . . 13
3 Generate Entity tool. . . . 19
4 Run Connector . . . 22
5 Connector Monitoring . . . 23
6 Connector Management . . . 23
7 Connector Config Parameters . . . 27
License . . . 32

Preface

Solace Database Connector * Getting Started * Resource Requirements for Self-Managed Micro-Integrations * Enabling Workflows == Configuring Micro Integration * Configuring the Datapath for Your Self-Managed Micro-Integration

Management and Monitoring Micro Integration

Managing and Deploying Your Self-Managed Micro-Integration

Configuration

- Configuring Self-Managed Micro-Integrations
- Workflow Configuration

Database Source Configuration Options

application-operator.yml

These configuration options are all prefixed by `solace-persistence.`:

Config Option	Type	Valid Values	Default Value	Description
`datasource`	`embedded config`	`see Spring Datasource Configuration`		The spring datasource configuration to read data
`datasource.maxDbRetry`	`int`	`int`	`3`	retry times to db query on error occurred
`datasource.dbRetryWaitMilliseconds`	`int`	`int`	`3000`	retry wait time to db query on error occurred, in millisecond
`datasource.incrementalRetryCycle`	`int`	`int`	`3`	Number of retry attempts before the retry wait time starts to increment. Retry wait time remains `dbRetryWaitMilliseconds` for retries less than this value.
`datasource.numberOfIncrementalTime`	`int`	`int`	`5`	Number of retry attempts before the retry wait time stop to increment. Retry wait time stop increment for retries get to this value.
`datasource.incrementalTime`	`int`	`int`	`3000`	time to incremental retry, in millisecond
`jpa`	`embedded config`	`see Spring JPA Configuration`		The spring jpa configuration
`source.sendBatchSize`	`int`	`send in batch`		currently only support sending one by one
`source.sendPoolSize`	`int`	`max 255`	`100`	the thread pool size used for sending messages

Config Option	Type	Valid Values	Default Value	Description
source.queryMax	int	query records a time	1000	query from table for the records number a time
source.triggerInterval	long	long	1000	interval of each db query, in millisecond
source.enableRetry	string	true or false	true	if true, will retry to do db query
source.retryInterval	int	int	5	interval of sending to solace on error occurred, in second
source.maxSendRetry	int	int	3	retry times of sending to solace on error occurred, in second
source.enableBufferedJpaBatch	string	true or false	true	if true, connector will write back indicator values in batch mode
source.bufferWaitSeconds	int	int	5	max time to wait for the batch write, in second
source.enableDatabaseTimestamp	string	true or false	true	if true, will include the reading DB record time in the message header.
source.enableConnectorTimestamp	string	true or false	true	if true, will include the connector sending message time in the message header.
source.enableBenchmarkInfo	string	true or false	false	if true, print the performance statistics.
source.dbQueryMode	string	jpa or sql	jpa	the internal implementation of db query
source.strategy	string	readingIndicator , timestamp or sequential	readingIndicator	the strategy of handling db rows. readingIndicator stands for reading data by indicator values, timestamp stands for reading data by timestamp order and range ,sequential stands for reading data by specified order row;

application-db-processor.yml

These configuration options are all prefixed by `solace-db.bindings.<binding-name>.consumer`:

Config Option	Type	Valid Values	Default Value	Description
tableKey	string			key name for a source table
tableClass	string			entity class reference of source table
childTableKey	string			key name for a source child table
childTableClass	string			entity class reference of source child table
topic	string	topic name with multiple levels, using /		data topic name, send the event of record to this topic
lvqTopic	string	topic name with multiple levels, using /		lvq topic name, send checkpoint data to this topic
parentChildRelationKey	string			parent-child relations mapping as: <parent_table_column1>:<child_table_column1>,<parent_table_column2>:<child_table_column2>
dbParentCollection	string	map object		map object in the parent entity
parentChildMode	string	one2many or many2one	one2many	the parent-child table relationship mode
dbReadIndicatorColumn	string	source table indicator column		source table read indicator column
sql	string	a select sql		self defined sql to query source table, `:<property-name>` for variable
dbTrackingId	string	tracking id value		specify the tracking ID column property, disabled when value not presented

Config Option	Type	Valid Values	Default Value	Description
dbReadTime	string	connector will update this column when reading this record		specify the read time column property, update this column with the reading record time by connector. disabled when value not presented
dbReadIndicatorColumnUpdateValueInProgress	string	indicator column will change to this value		after records read by connector and in the process, only available on strategy readingIndicator
dbReadIndicatorColumnUpdateValueProcessed	string	indicator column will change to this value		after records read by connector, sent to solace as event and got the ack, only available on strategy readingIndicator
dbReadIndicatorColumnUpdateDefaultValue	string	default value		initial value and ready to read by connector, only available on strategy readingIndicator
dbReadIndicatorColumnUpdateFailedValue	string	set to a failed value		error or failed during processing, only available on strategy readingIndicator
dbReadRecordSequentialIndicator	string	table column property		table records reading sequencial column property
dbReadRecordSequentialOrder	string	asc or desc	asc	read table records
dbReadIndicatorBridgeFailedValue	boolean	true or false	false	if true, it will read the failed ones after starting the connector, if failed, it will ignore the failed one.
dbReadIndicatorBridgeInProgressValue	boolean	true or false	false	if true, it will read the in-progress ones after starting the connector, if failed, it will ignore the progress one.

Config Option	Type	Valid Values	Default Value	Description
propertiesEnabled.dbTrackingId	boolean	true or false	none	if false , dbTrackingId property must be configured as empty value; if true, dbTrackingId property must be configured and not empty
propertiesEnabled.dbReadTime	boolean	true or false	none	if false , dbReadTime property must be configured as empty value; if true, dbReadTime property must be configured and not empty
customMessageHeaders	map	header name		it will have custom_header in the message property, its value is from the record of the column property specified, prefixed by MESSAGE_HEADER_

Database Sink Configuration Options

application-operator.yml

These configuration options are all prefixed by solace-persistence.:

Config Option	Type	Valid Values	Default Value	Description
datasource	embedded config	see Spring Datasource Configuration		The spring datasource configuration to write data
datasource.maxDbRetry	int	int	3	retry times to db query on error occurred
datasource.dbRetryWaitMillseconds	int	int	3000	retry wait time to db query on error occurred, in millisecond

Config Option	Type	Valid Values	Default Value	Description
datasource.incrementalRetryCycle	int	int	3	Number of retry attempts before the retry wait time starts to increment. Retry wait time remains dbRetryWaitMilliseconds for retries less than this value.
datasource.numberOfIncrementalTime	int	int	5	Number of retry attempts before the retry wait time stop to increment. Retry wait time stop increment for retries get to this value.
datasource.incrementalTime	int	int	3000	time to incremental retry, in millisecond
jpa	embedded config	see Spring JPA Configuration		The spring jpa configuration
sink.enableDMQRetry	boolean	true or false	false	if true, binder retry will be disabled, the solace broker will handle bad messages via DMQ
sink.jpaBatchMode	string	true or false	true	true to enable batch insert to the DB table
sink.jpaBatchSize	int	25,100…	25	Set the batch size to DB when jpaBatchMode is true
sink.jpaMaxRetry	int	2,3,4…	3	Retry times to insert to DB table times.
sink.jpaRetryWaitMilliseconds	int	3000	empty	Interval between retry, milliseconds
sink.redirectOnFailEnable	boolean	true or false	false	when true, if error occurred during consuming message payload , it will redirect this message to another new topic with a redirectPrefix prepend to the original topic name

Config Option	Type	Valid Values	Default Value	Description
sink.redirectOnUnknownTopic	boolean	true or false	false	when true, if the topic name in the queue is not mapped to any table, it will redirect this message to another new topic with a redirectPrefix prepend to the original topic name,available only when redirectOnFailEnable=true
sink.redirectTopicPattern	string		RD/${TOPIC}	pattern for redirected topic, ${TOPIC} stands for the original topic name
sink.payloadFormat	string	xml or json	xml	payload format sent to Solace queue

application-db-processor.yml

These configuration options are all prefixed by solace-db.bindings.<binding-name>.producer:

Config Option	Type	Valid Values	Default Value	Description
dbConnectorBlobEncoding	string	BASE64 or HEX	BASE64	Blob Encoding format in the message payload
dbConnectorTimestampFormat	string	timestamp format	yyyy/dd/MM.HH:mm:ss	Set the timestamp format pattern
dbMsgHeaderTimestampFormat	string	timestamp format	yyyy/dd/MM.HH:mm:ss	Set the message header timestamp format pattern
entities	map			map of embedding configurations of entity

ParentChildRelation configuration options are all prefixed by solace-db.bindings.<binding-name>.producer.parentChildRelation:

Config Option	Type	Valid Values	Default Value	Description
childNode:	string			the json node name relative to jsonNodePrefix conatins the child table data
childKey:	string			name of child entity if needed, childKey should be specified in entities

Config Option	Type	Valid Values	Default Value	Description
parentChildRelationKey:	string			parent-child relations mapping as: <parent_table_column1>:<child_table_column1>,<parent_table_column2>:<child_table_column2>

Entity configuration options are all prefixed by `solace-db.bindings.<binding-name>.producer.entities.<entity_name>`:

Config Option	Type	Valid Values	Default Value	Description
clazz:	string			entity class reference
jsonNodePrefix:	string			json node path of entity data
topicOperations:	map			map of embedding configurations of topic operation detail mappings

Topic operation configuration options are all prefixed by `solace-db.bindings.<binding-name>.producer.entities.<entity_name>.topicOperations`:

Config Option	Type	Valid Values	Default Value	Description
save.topics	list			list of topic names apply for JPA save operation, the event of contain this topic name will be saved to the table. wildcard >,* can be used for filter the topic.
save.jsonNodePrefix:	string	eg. node/payload		json node path of entity data for save operation
delete.topics	list			list of topic names apply for SQL delete operation, the event of contain this topic name will be saved to the table. wildcard >,* can be used for filter the topic.
delete.jsonNodePrefix:	string	eg. node/payload		json node path of entity data for delete operation

More information

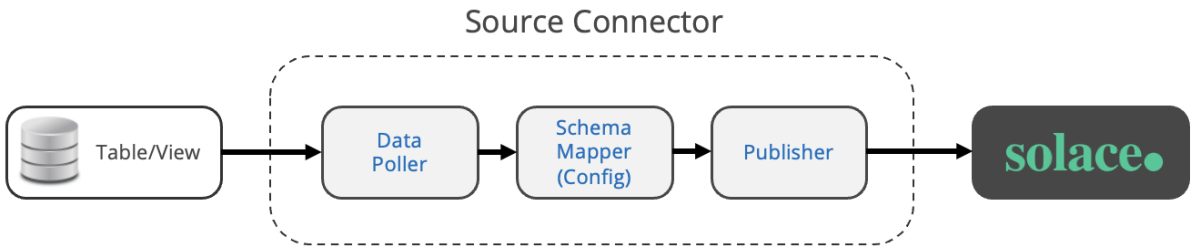
Solace Pubsub Connector For Database Source

1 Introduction

The **Solace Pubsub Connector For DB Source** has been built to make DB transactions participated in event mesh by using the Connector for DB Source

Solace Pubsub Connector For DB Source helps to read the database records and make it available as an event in the Solace Event Mesh for enterprise applications to consume.

Note: Pulls the records which are marked as unread.



1.1 Purpose

To guide the technical teams to setup the **Solace Pubsub Connector For DB Source**.

1.2. Scope

Covering the step-by-step guide to setup the **Solace Pubsub Connector For DB Source**.

1. Generate JPA Entities using the tool
2. Configure DB Souce Connector
    1. Connection Configuration (Database & Solace)
    2. Connector & Entity Configuration
    3. Schema Mapping (Table Structure to Solace Event Payload)

`Note: This document currently captures the deployment in Java environment.`

1.3 DB Connector Features

S.No	Feature
1	The source DB object from where Adapter polls remains as-is (as EMS). No change expected at application DB end (Includes Tables, Views & Child Table)
2	Adapter should pick records in sequence only (Based on Sequence Number and timestamp field)
3	The Adapter should be able to update each record with appropriate flag (in existing adb_l_delivery_status field) N - New : set by trigger P - Pending: under process (picked up by adapter, not yet published to messaging layer) C - Completed: Processing completed (published successfully to messaging layer) F - Failed: Fail to publish record
4	The adapter should have the capability to read in batch and update flag in batch to avoid chattiness and ensure better performance
5	Oracle DB should be supported (12g onwards (19C and ADG and RAC)
6	Design should be future proof for oracle version upgrades. Should support Thick and Thin DB driver
7	Should support ADG URL
8	Should have option to retry before marking record as 'F' The query to poll should be configurable.
9	Schema mapping feature should be Forward compatible.
10	Any new addition to DB schema should be ignored by Adapter unless Field name and mapping is provided.
11	Option to specify the data type of each field in the mapping.
12	Support for CLOB and BLOB
13	Date format to be as per the TIBCO specs - Should be configurable. default is yyyy-MM-dd’T’HH:mm:ss.SSS
14	Logging & Syslog

S.No	Feature
15	High Available & Fault Tolerance
16	Event Headers: In Solace Message Header, for each message published, add Tracking Id coming from source table Time at which picked from DB Time of publish to Solace

2 DB Connector Configuration

2.1. Pre-requisites

- JDK 17
- Maven 3.8+
- Solace
- Database for Oracle or MySQL

2.2. Install Solace Framework and Transform Engine

Go to the Framework jar and pom files folder and run below command to install into your local maven repo:

```
mvn install:install-file
-Dfile=solace-transformation-engine-1.0.10-SNAPSHOT.jar
-DgroupId=com.solace.connector.core
-DartifactId=solace-transformation-engine  -Dversion=1.0.10-SNAPSHOT
-DpomFile=solace-transformation-engine-1.0.10-SNAPSHOT.pom
```

2.3. Solace Pubsub Connector For DB Source

Download the Solace DB connector (spring-cloud-stream-binder-db and pubsubplus-connector-database) and unzip to the location/machine where the connector is getting installed

Go to spring-cloud-stream-binder-db folder and run:

```
mvn -DskipTests=true install
```

Go to pubsubplus-connector-database folder and run:

```
mvn -DskipTests=true clean package
```

2.4 DB Configuration

Cofiguration File: `projConfig/application.yml`

```
solace-persistence:
  datasource:
    driver-class-name: oracle.jdbc.OracleDriver
    url: jdbc:oracle:thin:@//192.168.3.178:1524/ORCLCDB
    username: c##test
    password: test
    hikari:
      initializationFailTimeout: -1
      connection-timeout: 5000
  jpa:
    hibernate:
      ddl-auto: none
      show-sql: false
    properties:
      hibernate:
        jdbc: batch_size: 100
        order_updates: true
        generate_statistics: false
      dialect: org.hibernate.dialect.Oracle10gDialect
      database: oracle
  source:
    dbQueryMode: jpa # jpa or sql(not fully supported)
    sendBatchSize: 1 #records per message
    sendPoolSize: 100 # send thread pool size, defalut 100 max 255
    queryMax: 2 #limited records per DB query
    maxDbRetry: 3
    triggerInterval: 2000
    retryInterval: 10 # seconds
    enableRetry: true #enable retry to send the message to Solace.
    retryCount: 3
    enableBufferedJpaBatch: true # only available when sendBatchSize = 1
    bufferWaitSeconds: 5
    enableTrackingId: true #update tracking Id to table
    enableDatabaseTimestamp: true
    enableConnectorTimestamp: true
    enableBenchmarkInfo: true
    skipMismatchError: false
    indicatorFlag: true
```

- Fill in all the required details of `solace-persistence.datasource` to establish a connection with Database.
- `solace-persistence.datasource` section, `batch_size`\: Given batch size will allow connector to update the status back to Database in a batch mode
- `solace-persistence.datasource.hikari` section, Hikari Settings: Will enable connector to perform retry/reconnect until it is timeout.
- `solace-persistence.source` section, Additional configurations to pull the data from Database.

2.5 Solace Configuration

Configuration File: `projConfig/application.yml`

```
spring:
  cloud:
    stream:
      bindings:
        input-0:
          destination: souceBinderLVQ #LVQ queue name
          binder: solace-db #Solacedatabase binder
          group: no
        output-0:
          destination: dxb/passenger #topic to send data
          binder: solace
          content-type: application/x-java-serialized-object
```

- `spring.cloud.stream.bindings.input-0` section, Fill in all the required details to establish a connection with Solace
- `spring.cloud.stream.bindings.output-0` section, Set contentType to application/x-java-serialized-object to send plain text message, the default is bytes.

Configure connection to Solace:

```
  java:
    host: tcp://192.144.217.87:55555
    msgVpn: default
    clientUsername: admin
    clientPassword: admin
    connectRetries: -1
    reconnectRetries: -1
    apiProperties:
      pub_ack_window_size: 50
      pub_ack_time: 200
```

Configure the management part for the HA:

```
solace:
  connector:
    workflows: # Workflow configuration
      0:
        enabled: true  # If true, the workflow is enabled.
        acknowledgment:
          publish-async: true
        transformation:
          mappingFile: ""
          mode: customentitytoxml
    #      1:
```

```
    #        enabled: false  # If true, the workflow is enabled.
    #        acknowledgment:
    #          publish-async: true
    management:
      leader-election:
        mode: active_standby  # The connector's leader election mode. (values:
standalone, active_active, active_standby)
        fail-over:
          max-attempts: 3  # The maximum number of attempts to perform a fail-over.
          back-off-initial-interval: 1000  # The initial interval (milliseconds) to
back-off when retrying a fail-over.
          back-off-max-interval: 10000  # The maximum interval (milliseconds) to back-
off when retrying a fail-over.
          back-off-multiplier: 2.0  # The multiplier to apply to the back-off interval
between each retry of a fail-over.
      queue: management-queue-push  # The management queue name.
      session: # The management session. This has the same interface as that used by
`solace.java.*`. For more info: https://github.com/SolaceProducts/solace-spring-
boot/tree/master/solace-spring-boot-starters/solace-java-spring-boot-starter#updating-
your-application-properties
        host: tcp://192.144.217.87:55555
        msgVpn: default
        client-username: admin
        client-password: admin
```

2.6. Config file and mapper file

Configuration File: `connector_config.properties\`:

```
#table entity mapping
source_passenger=entity.source.database.com.solacecoe.connectors.SourcePassenger
passAddress=entity.source.database.com.solacecoe.connectors.PassAddress

DB_TABLE=source_passenger
#--------------------source_passenger-------------
#<DB_TABLE>.ENABLE_<property>=true or false, if false , target property must be
configured as empty value; if true, target property must be configured and not empty
source_passenger.ENABLE_DB_READ_TIME=true
source_passenger.ENABLE_ DB_TRACKING_ID=false
#use default output-0
source_passenger.BINDING=output-0
#data topic name, send the event of record to this topic
source_passenger.TOPIC=dxb/source_passenger
#lvq topic name, send checkpoint data to this topic
source_passenger.LVQ_TOPIC=lvq/source_passenger
#parent table name
source_passenger.DB_TABLE=source_passenger
#DB_PARENT_CHILD_MODE: many2one or one2many, default is one2many
source_passenger.DB_PARENT_CHILD_MODE=many2one
#key column property in the parent table, have relationship with child table key.
```

```
source_passenger.PARENT_CHILD_RELATION_KEY=passengerId:passId,
nationality:zip,<parentKey>:<childKey>
#map in the main entity.
source_passenger.DB_PARENT_COLLECTION=addresses
#child table name
source_passenger.DB_CHILD_TABLE=passAddress
#source table read indicator column
source_passenger.DB_READ_INDICATOR_COLUMN=flag
#specify the tracking ID column property
source_passenger.DB_TRACKING_ID=dbtrackid
#specify the read time column property, update this column with the reading record
time by connector
source_passenger.DB_READ_TIME=readtime
# status of the indicator column value.
source_passenger.DB_READ_INDICATOR_COLUMN_UPDATE_VALUE_INPROGRESS=p
source_passenger.DB_READ_INDICATOR_COLUMN_UPDATE_VALUE_PROCESSED=c
source_passenger.DB_READ_INDICATOR_COLUMN_DEFAULT_VALUE=n
source_passenger.DB_READ_INDICATOR_COLUMN_FAILED_VALUE=f
# sequeial column property
source_passenger.DB_READ_RECORD_SEQUENTIAL_INDICATOR=id.passengerId
# order by, asc or desc
source_passenger.DB_READ_RECORD_SEQUENTIAL_ORDER=asc
#if true, it will read the failed ones after starting the connector, if failed, it
will ignore the failed one.
source_passenger.DB_READ_INDICATOR_BRIDGE_FAILED_VALUE=true
##message heasers
source_passenger.ENABLE_MESSAGE_HEADER_TRACK_ID=true
source_passenger.MESSAGE_HEADER_TRACK_ID=dbtrackid
#MESSAGE_HEADER_CREATE_TIME=createTime
```

Create mapper files for source_<table_name>_trgt_schema_mapper.yml for each table, ex source_passenger_trgt_schema_mapper.yml</table_name>

```
solace-connector-mapper:
  mapper-metadata:
    escape-special-characters: true
    schema-type:
    encrypted:
  payload-mapper:
    mapper-name0:
      source-xpath: source_passenger/passengerId
      source-datatype:
      target-xpath: passenger/id
      target-datatype:
    mapper-name1:
      source-xpath: source_passenger/name
      source-datatype:
      target-xpath: passenger/pname
      target-datatype:
    mapper-name2:
```

```
      source-xpath: source_passenger/birthday
      source-datatype:
      target-xpath: passenger/birthday
      target-datatype:
    mapper-name3:
      source-xpath: source_passenger/passAddress/city
      source-datatype:
      target-xpath: passenger/addresses/passAddress/city
      target-datatype:
```

2.7 Connector Logging

Logback has been used to enable the logging mechanism.

Configuration File: `resourceslogback.xml`

You can change your log default folder in logback.xml:

```
<property name="LOG_HOME" value="log/pull-connector/" />
```

You can change your log level as showed below:

```
<!--DEBUG,INFO,WARN,ERROR,FATAL-->

<root level="INFO">
  <appender-ref ref="STDOUT" />
  <appender-ref ref="FILE" />
  <appender-ref ref="RSYSLOG" />
</root>
```

2.8 Admin client configuration

Please configure the admin sever as follows in application.yml

```
spring:
  boot:
    admin:
      client:
        enabled: true
        url: http://localhost:8082       #configure your admin server url
        auto-registration: true
  application:
    name: Instance006                    #configure your application name
```

3 Generate Entity tool.

Hibernate API's has been used to generate the JPA entities and load into the connector code. This will reduce the manual efforts to write the entity file for all the tables.

Note: We are also finding a way to automate the entire entity generation process to reduce the manual interventions.

Download the generateEntity project and edit the below show file to generate the entities on the connected schema.

`pom.xml` configuration

```
<configuration>
    <!--
<templatePath>${project.basedir}/src/main/resources/jpagen/tempalate/</templatePath>--
>
    <!-- Defaults: -->
    <outputDirectory>${project.basedir}/src/main/java/</outputDirectory>
    <!--hibernate configuration file-->
    <propertyFile>src/main/resources/hibernate.properties</propertyFile>
    <!--DB and Entity data type mapping configuration file-->
    <revengFile>src/main/resources/reveng.xml</revengFile>
    <!--entity operation info -->
    <revengStrategy></revengStrategy>
    <packageName>com.solace.connector.db.pull.entity</packageName>
    <detectManyToMany>true</detectManyToMany>
    <detectOneToOne>true</detectOneToOne>
    <detectOptimisticLock>true</detectOptimisticLock>
    <createCollectionForForeignKey>true</createCollectionForForeignKey>
    <createManyToOneForForeignKey>true</createManyToOneForForeignKey>
    <!-- true JPA anotation -->
    <ejb3>true</ejb3>
    <jdk5>true</jdk5>
</configuration>
```

DB configuration The file `hibernate.properties` location is configured in pom.xml file:

```
default\_schema=TEST
hibernate.connection.driver\_class=oracle.jdbc.driver.OracleDriver
hibernate.connection.username=system
hibernate.connection.password=oracle
hibernate.connection.url=jdbc:oracle:thin:@localhost:47161/xe
hibernate.dialect=org.hibernate.dialect.Oracle10gDialect
```

Data type mapping configuration The file `reveng.xml` location is configured in pom.xml file:

```
<sql-type jdbc-type="DATE" hibernate-type="java.util.Date"/>
    <sql-type jdbc-type="TIMESTAMP" hibernate-type="java.util.Date"/>
```

```
    <!--<sql-type jdbc-type="NUMERIC" hibernate-type="java.lang.Double"/>-->
    <sql-type jdbc-type="DECIMAL" hibernate-type="java.lang.Double" />
    <sql-type jdbc-type="CHAR" hibernate-type="java.lang.String" />
    <sql-type jdbc-type="NUMERIC" length="10" precision="3"  hibernate-
type="java.math.BigDecimal" />
    <sql-type jdbc-type="NUMERIC" length="20" precision="5"  hibernate-
type="java.math.BigDecimal" />
    <sql-type jdbc-type="NUMERIC" length="37" precision="4"  hibernate-
type="java.math.BigDecimal" />
    <sql-type jdbc-type="FLOAT"   hibernate-type="java.math.BigDecimal" />
    <sql-type jdbc-type="NUMERIC" length="38" precision="0"  hibernate-
type="java.math.BigDecimal" />
    <sql-type jdbc-type="NUMERIC"  hibernate-type="java.math.BigDecimal" />
    <sql-type jdbc-type="BLOB"   hibernate-type="java.lang.Byte[]" />
    <sql-type jdbc-type="CLOB"   hibernate-type="java.lang.String" />
```

Note:

- match-name could be a single table name, or it can be a wildcard string (.*) to generate the entity.
- Some of the minute modifications needed to fit to DB Connector entity requirements
- Also, no modification to single table entity, this is only for parent/child table.
- Add tablename to generate a single table

```
<table-filter match-schema=".*" match-name="tablename" />
```

Parent table ex:

```
// Below are generated by tool, should be removed:
@OneToMany(fetch=FetchType.LAZY, mappedBy="sourcePassenger")
private Set<PassAddress> passAddresses = new HashSet<PassAddress>(0);
Set<PassAddress> passAddresses
Set<PassAddress> getPassAddresses()
Set<PassAddress> passAddresses

Replace all the Set<> to be List<Map> in the entity:
private List<Map> addresses;
@JsonIgnore
@Transient
public List<Map> getAddresses() {
    return addresses;}
public void setAddresses(List<Map> addresses) {
    this.addresses = addresses;}
```

The `addresses` is the `DB_PARENT_COLLECTION` `of` `connector_config.properties` .

If there is no PK for parent table. 2 entities are generated for parent table, `entity` and `entityId`.

Move all the properties/methods from entityId to entity.

In the entity, remove `@AttributeOverrides({})`.

For example: Parent table entity: `SourcePassenger` Parent table entityId: `SourcePassengerId`

Move all the property in SourcePassengerId to SourcePassenger, and remove `@AttributeOverrides({})` in SourcePassenger and add below:

Add `@Id` on the unique column of `SourcePassenger`\:

```
@Id
@Column(name="PASSENGER_ID", nullable=false, precision=10, scale=0)
```

Child table ex:

If there is no PK for child table. 2 entities are generated for child table, `entity` and `entityId`. Move all the properties/methods from entityId to entity .

In the entity, remove `@AttributeOverrides({})`.

For example: Child table entity: `PassAddress` Child table entityId: `PassAddressId` Move all the properties in PassAddressId to PassAddress. Add @Id on the unique column of PassAddressId:

```
@Id
@Column(name="ID", nullable=false, precision=10, scale=0)
```

Create VIEW entity:

When it is a VIEW object in DB, please treat it as a single table, just point your table name to your view name.

For example, if you have a view `Customer`. 2 entities will be generated, `Customer` and `CustomerId`, please move all properties from CustomerId to Customer. In the entity, remove `@AttributeOverrides({})`. Add `@Id` on the unique column of `Customer`\:

```
@Id
@Column(name="CUSTOMER_ID", nullable=false, precision=10, scale=0)
```

Command to run:

```
mvn hibernate-tools:hbm2java
```

Generate the entity jar:

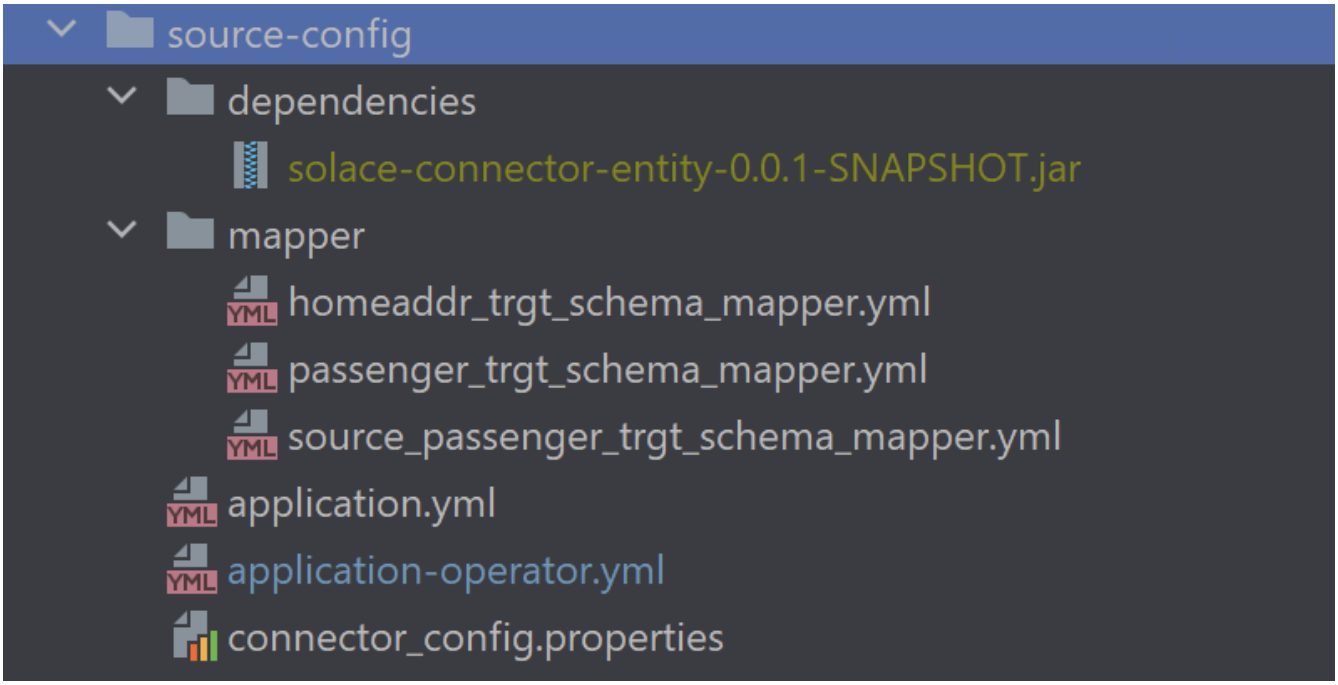
Put the entities to your entity project, manually add entity repo and generate the entity jar, put the entity jar to \\config\\dependencies folder.

4 Run Connector

4.1 Command line to run the Solace Pubsub Connector For DB source

Go the project

```
mvn clean package
```



Go to project folder and run as standalone mode:

```
java -cp ./target/pubsubplus-connector-database-1.2.0-SNAPSHOT.jar
-D"loader.path=./source-config/dependencies,./source-config"
org.springframework.boot.loader.PropertiesLauncher
--spring.config.additional-location="./source-config/application-operator.yml"
-server.port=8081
```

4.2 Command line to run the Solace Pubsub Connectors as HA

Please specify server.port to another value that is not used.

```
java -cp ./target/pubsubplus-connector-database-1.2.0-SNAPSHOT.jar
-D"loader.path=./source-config/dependencies,./source-config"
org.springframework.boot.loader.PropertiesLauncher
--spring.config.additional-location="./source-config/application-operator.yml"
-server.port=8082
```

Please refer to 1.4.2 Solace Configuration> Configure the management part for the HA.

4.3 Deploy as docker container and run

```
docker run --name pubsubplus-connector-database -v
D:\\GitHub\\docker\_config:/config pubsubplus-connector-database:v1
docker\_config refers to source-config and sink-config
```

5 Connector Monitoring

Connector monitoring is a Spring Boot Admin server which helps to visualize the connectors deployed in the organization

5.1 Prerequisites

- Maven 3
- JDK 17

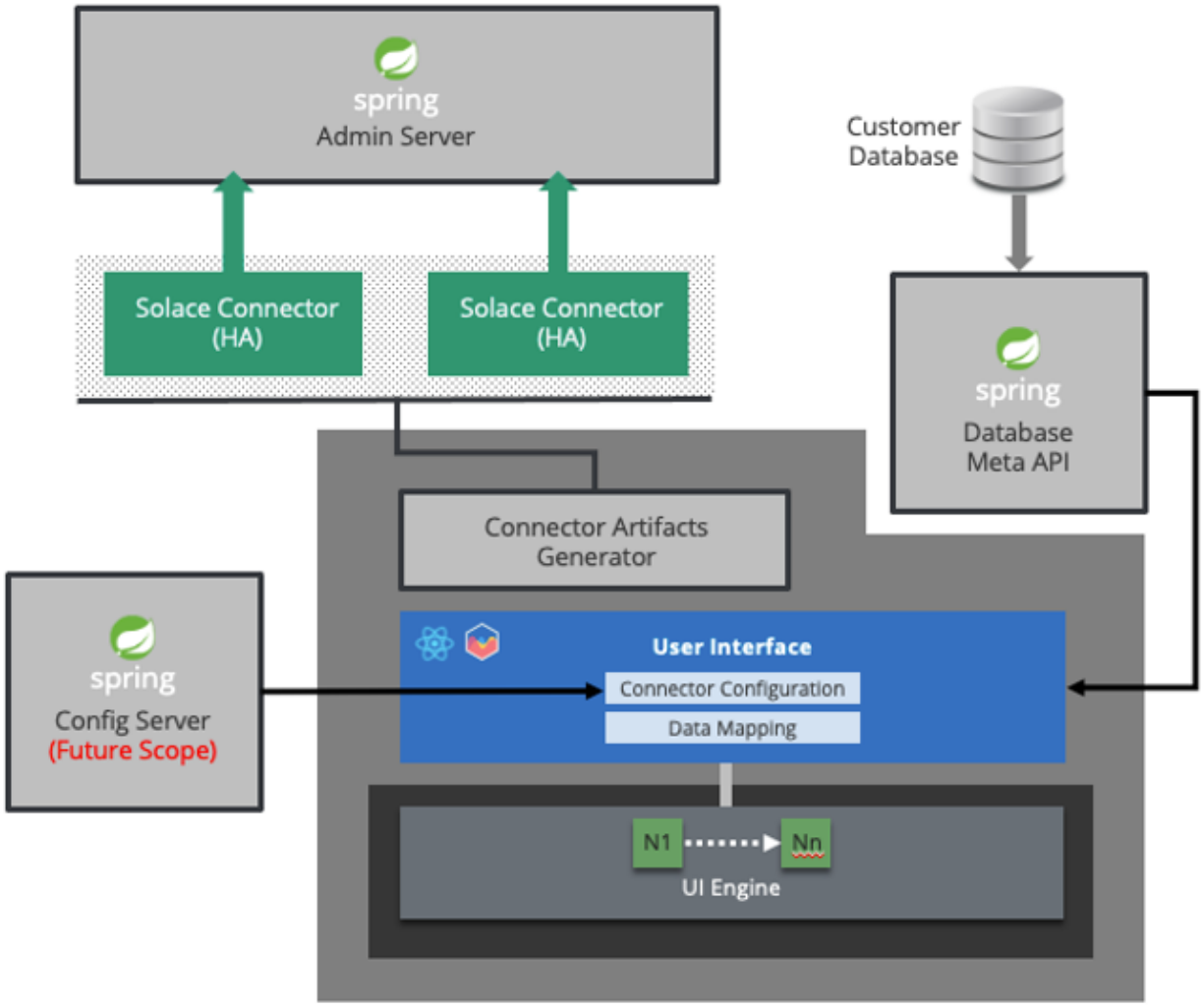
5.2 Infrastructure Requirement

Component Name	Cores	RAM (GB)	Runtime
Spring Boot Admin	2	4	Java

6 Connector Management

- Connector management is a UI based tool to generate the code binary based on the configuration provided in the tool.
- All the connector configurations will be stored in the database attached to the CM
- Will allow to edit the configuration for the available connectors in the CM database.

6.1 Architecture



6.2 Pre-Requisites

S.No	Runtime	Version	Description
1	NodeJS	V12	Used for CM runtime
2	NPM	V6	Used to manage node packages required by CM
3	PM2	5.2	Used to manage NodeJS process for scalabilty
4	Java	17	Used for Database Meta API runtime
5	Oracle Instant Client	21	Required for NodeJS library to initiate connection with Oracle
6	Maven	3	Used to Manage dependencies required by Meta API

6.3 Infrastructure Requirement

Solace Connector Management will be deployed in a single VM

Component Name	Cores	RAM (GB)	Runtime
Connector Management (UI+Services)	4	6	NodeJS
DB Meta API (Per Schema)	1	2	Java

6.4 Connector Management Installation Guide

CM Deployment kit contains:

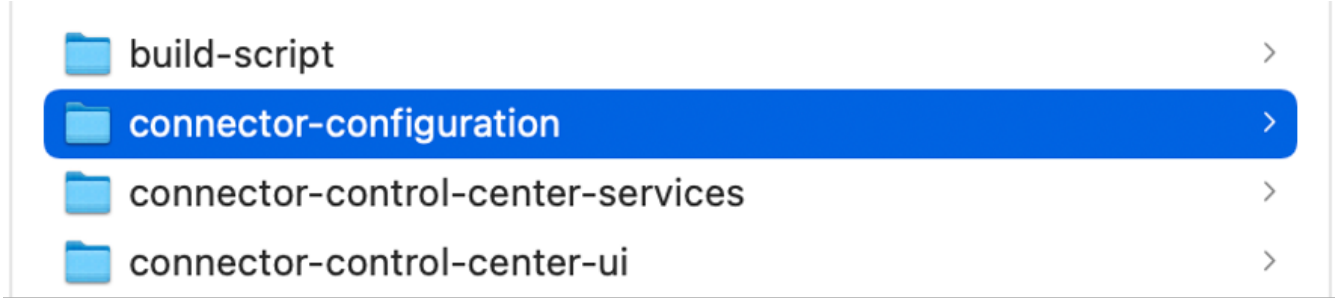
- Schema script (Oracle)
- This script includes all the required database commands to create tables required for Connector Management
- Spring Admin Server Binary
- This binary spins ups the Spring Admin server after providing required configurations
- CM NodeJS Binary
- User interface component which can be used to configure/manage connectors.

Database preparation

- Create Schema: `connector_control_schema`
- Create the tables: Execute the Schema Script provided in the kit to create the required tables.

Deploy CM:

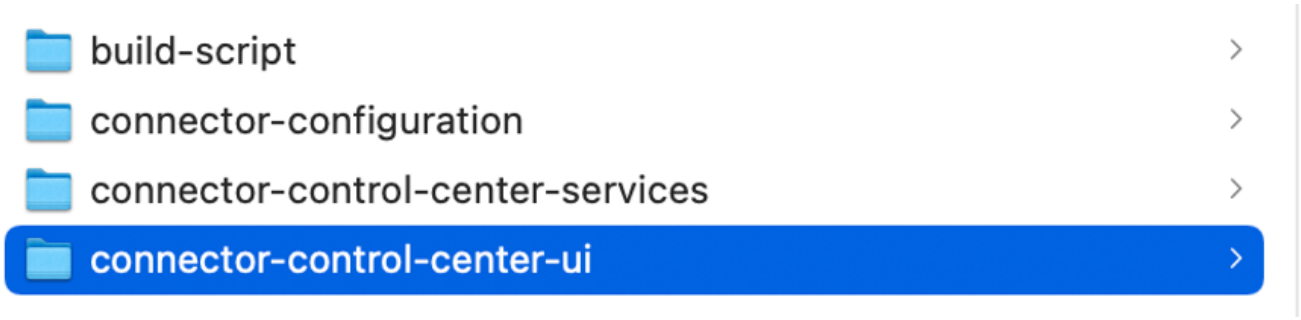
- Unzip the file <file-name> to the desired location</file-name>
- Edit the ecosystem.config.js file located in `connector_configuration` folder. This configuration is used by PM2



```
env_dev: {
  "NODE_ENV": "dev",
  "DATABASE_HOST": "",
  "DATABASE_PORT": 5432,
  "DATABASE_USER": "",
  "DATABASE_PASSWORD": "",
  "DATABASE_NAME": "",
  "CONTROL_CENTER_UI_DOMAIN": "",
  "CONTROL_CENTER_BACKEND_DOMAIN": "",
  "SECRET_KEY": "",
  "REFRESH_TOKEN_KEY": "",
  "API_PORT": 3001,
  "SOLACE_VPN_NAME": '',
  "SOLACE_HOST": '',
  "SOLACE_AUTH_TOPIC": "",
  "SOLACE_ACC_DATA_PROTOCOL": "",
  "SOLACE_ACC_REST_PORT": 9443,
  "SOLACE_CLIENT_USER": "",
  "SOLACE_CLIENT_PWD": "",
  "CCC_GENERATE_ENTITY_TEMPLATE_PATH": 'generateEntity.zip',
  "CCC_SOLACE_DB_PULL_CONNECTOR_TEMPLATE_PATH": 'solace-db-pull-connector.zip',
  "CCC_DOWNLOAD_GENERATE_ENTITY_PATH" : "deploy-connectors"
},
```

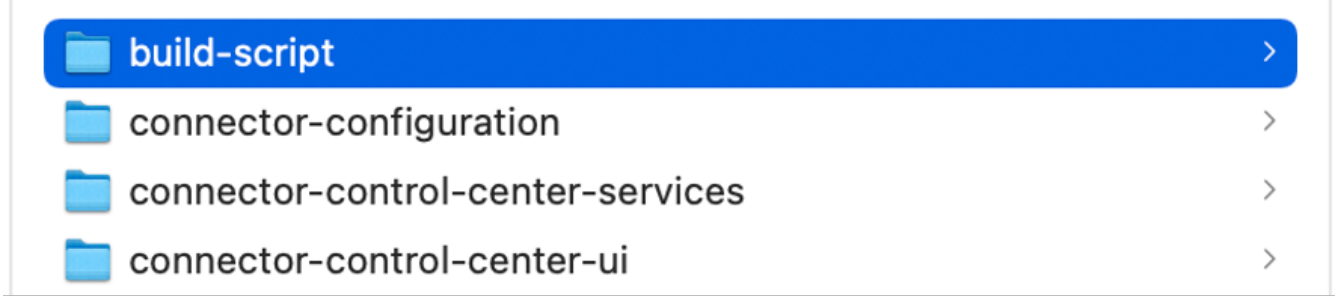
The same configuration can be repeated for different environments like dev, qa, prod.

Edit the `.env` file in the UI folder `connector-control-center-ui`




```
REACT_APP_API_SERVICES_URL = 'http://<host-name>:3001'
REACT_APP_DOMAIN = 'http://<host-name>:3000'
#### CCC META API
REACT_APP_CCC_META_API_URL = 'http://<host-name>:8082'
```

Run the script located in `build-script` folder in the kit to create the build.



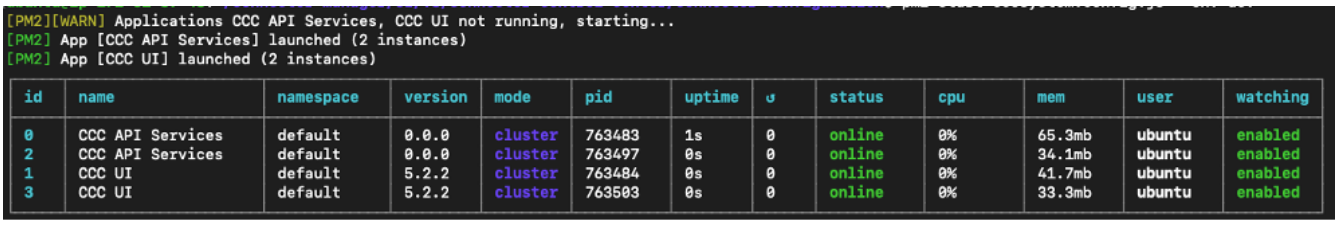
Command to execute the script

```
cc_deploy.sh
```

Once successfully executed the script navigate to connector-configuration folder and execute the below command

```
pm2 start ecosystem.config.js env <dev,qa,prod>
```

To validate, run `pm2 ls` and you should see the following services up and running.



6.5. Connector Generation Guide

Generating the connector binary is a three-step process

1. Add schema to CM
2. Generate entity (JPA Notation) for the schema
3. Generate connector configurations as per the selected schema tables to bridge the data to Solace as events

6.6. Add Schema

6.7. Generate Entity

6.8. Generate Connector Configurations

7 Connector Config Parameters

7.1. Connector parameters

Config Type	Config Location	Parameter (Hierarchy)	Description
Generic-DB	application.yml	solace-persistence:datasource:driver-class-name	DB Connection Details
solace-persistence:datasource:url			
solace-persistence:datasource:username			
solace-persistence:datasource:password			
solace-persistence:datasource:hikari:minimum-idle	Config the connection pool for DB		
solace-persistence:datasource:hikari:idle-timeout			
solace-persistence:datasource:hikari:maximum-pool-size			
solace-persistence:datasource:hikari:max-lifetime			
solace-persistence:datasource:hikari:connection-timeout			
solace-persistence:jpa:hibernate:ddl-auto	hibernate property, recommended to use none in production		
solace-persistence:jpa:show-sql	True, will print the queries in the logs		
solace-persistence:jpa:database	Type of the target database ex: mysql,oracle		

Config Type	Config Location	Parameter (Hierarchy)	Description
solace-persistence:jpa:properties:jdbc:batch_size	The size of jpa batch opration		
spring:cloud:function:stream:poller:fixed-delay	Defines the interval frequency of the data polling from Database		
spring:cloud:function:stream:bindings:dataCollectting-out-0:destination	Allowed to configure both static and dynamic topics Some of the parameterized notations will add data to topic during the runtime. {COL:column name}: adds the value for defined column name into the to the topic {DB_TABLE}: adds a table name to the topics based on connector_config.properties file Ex: DXB/{COL:passengerId}/{COL:name}/SGP/{DB_Table}		
logging	logback.xml	<root level="INFO"> <appender-ref ref="STDOUT" /> <appender-ref ref="FILE" /> <appender-ref ref="RSYSLOG" /> </root>	used logback to config logging, it has different levels: DEBUG,INFO,WARN,ERROR,FATAL
Source property	application.yml	solace-persistence:source:jpaBatchMode	Allow to merge multiple records into a single event
solace-persistence:source:queryMax	Set the max records for each query read from database		
solace-persistence:source:maxDbRetry	it will do retry to connect the db		

Config Type	Config Location	Parameter (Hierarchy)	Description
solace-persistence:source:enableRetry	If true, it will retry query to db		
solace-persistence:source:retryInterval	Messages resent to Solace in the defined interval		
solace-persistence:source:retryCount	Max retry count and mark the record as a failure attempt		
solace-persistence:source:enableBufferedJpaBatch	if true, connector will write back indicator values in batch mode		
solace-persistence:source:bufferWaitSeconds	timeout for getting ack from Solace		
solace-persistence:source:enableRetry	True, to enable resend the message to Solace		
solace-persistence:source:retryCount	When enableRetry is true, retry numbers to the Solace.		
solace-persistence:source:enableBufferedJpaBatch	True to enable batch update to the DB		
solace-persistence:source:bufferWaitSeconds	Interval between batch update to the DB		
solace-persistence:source:enableTrackingId solace-persistence:source:enableDatabaseTimestamp solace-persistence:source:enableConnectorTimestamp	True, to enable message header contain those info.		
solace-persistence:source:enableBenchmarkInfo	True, to print performance detail log		
Admin Client	application.yml	sping.boot.admin.client.enable:true	True to enable.

Config Type	Config Location	Parameter (Hierarchy)	Description
sping.boot.admin.client.enable:url	Set the admin server url		
sping.boot.application.name	Name of the application		
Solace	application.yml	sping.cloud.stream.binding.input-0.destination	LVQ queue name
sping.cloud.stream.binding.output-0.destination	LVQ topic name		
sping.cloud.stream.binding.output-0.content-type	Message content typ: plain text is application/x-java-serialized-object		

7.2 File Naming Convention

File Type	Naming Convention	Description	Note
Spring Application	application.yml	A standard spring boot configuration	Follow the sring standard yaml notations
Connector for DB source(Config)	connnector_config.properties	Solace DB Connector Configuration file	Generated
Connector For DB souce(Mapper File)	<DB_TABLE>_trgt_schema_mapper.properties </DB_TABLE>	A mapper file which will have the mapping definition between the parent table Schema to the target event payload schema	DB_TABLE value can be identified from connector_config.properties file. All the field mappings should follow the xpath format
Connector for DB source(Mapper File)	<child table="">_trgt_schema_mapper.properties</child>	A mapper file which will have the mapping definition between the parent table Schema to the target event payload schema	Generated

License

This project is licensed under the Apache License, Version 2.0. - See the `LICENSE` file for details.