Kafka Producer Design

ReD Merchant Fraud

Exported on 06/05/2020

Table of Contents

Document Revision History

Version	Author	Date Distriuted	Description of Changes
1.0	Venkata Praveen Jalem	03/13/2020	Initial Draft
1.1	Venkata Praveen Jalem	04/08/2020	Updated after reviews and agreements

Supporting Documents/References

Document	Location
Jiras	✓ MERF-22042[1] - Decouple Kafka writes from shared memory - Detail Design CLOSED ✓ MERF-22450[2] - Decouple Kafka writes from shared memory - Producer changes CLOSED
HLD for decoupling Kafka writes from FE Shared Memory	Decouple Kafka writes from FE shared memory - DRAFT[3]

1 https://jira.aciworldwide.com/browse/MERF-22042?src=confmacro
2 https://jira.aciworldwide.com/browse/MERF-22450?src=confmacro
3 https://wiki.aciworldwide.com/display/ReD/Decouple+Kafka+writes+from+FE+shared+memory+-+DRAFT

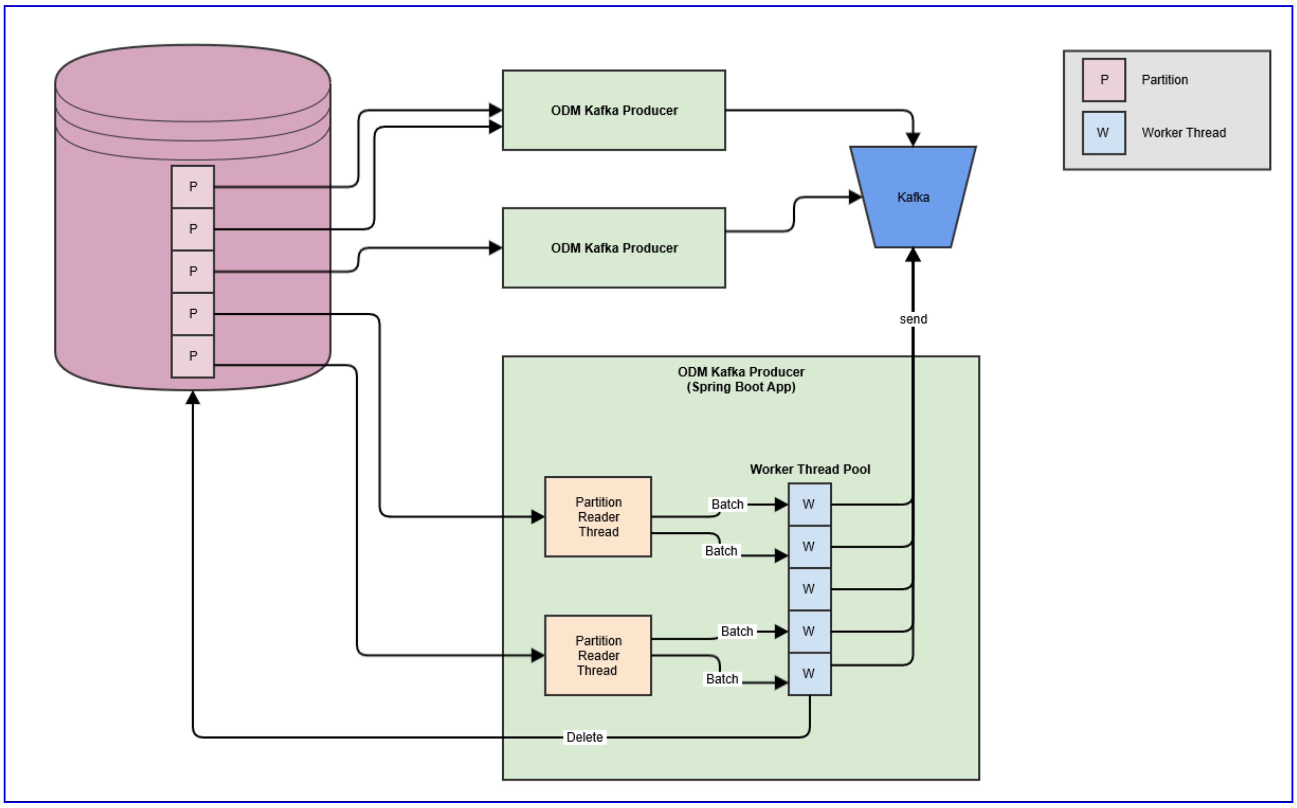
1 Overview

This page details the design for the new decoupled Kafka producer.

2 High Level Flow

2.1 Assumptions

- A single KAFKA_STAGE row holds all the data required to parse and send a transaction to Kafka.
- A CLOB field holds the transaction data in the same format as the message sent to socket in the current process.
- Consumers of the Kafka topic can handle resends/duplicates, in case of the application crash.



Kafka Producer is a standalone proxy application that reads the transaction data from the KAFKA_STAGE partitions and sends them out to Kafka. This is a scalable application and can be configured to process a given set of partitions. Each instance of the application would process a unique set of partitions. In the case where only a single instance of this application is run, all the partitions should be assigned to this instance.

This would be created as a standalone Spring Boot application to manage the object lifecycle and configuration better.

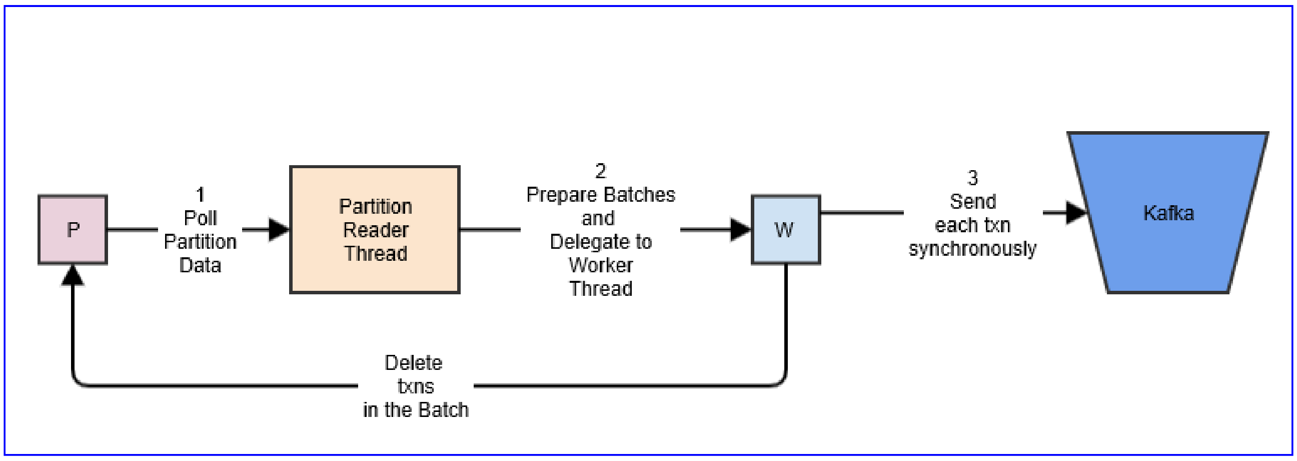
2.2 Partition Reader Thread

A Partition Reader Thread is responsible for handing transaction within a single partition. Each partition is polled by a separate thread of its own. It queries the records in the KAFKA_STAGE partition, creates batches of records and dispatches them to the worker threads for further processing. The query would return the oldest records, N records at a time, with N being configurable.

2.3 Worker Thread

A Worker Thread takes a batch of records, parses each record and transform it into a RedShieldTransaction and send it to Kafka synchrounously. On success, the handler would delete the record from the partition. On Failure the error would be logged and no action would be taken. The thread would be released back to the to the pool after processing all the records in the batch. Reader Thread would pick the failed records in the subsequent poll.

The diagram below explains the data flow between the components.



2.4 Kafka

A single KafkaProducer instance would be created for the entire application. It would be used by all the worker threads to send out the messages to the given topic. The producer would be configured with appropriate batch.size and linger.ms to achieve the required throughput.

Multiple instances of the KafkaProducer would be considered if a single instance is unable to handle the load. However, it is suggested that a single KafkaProducer instance can handle the load.

3 Resiliency

Status field based approach was considered originally to handle resiliency. However, it was rejected due to additional load it would impose on the DB with frequent updates. Following approach for resiliency is finalized for the initial producer implementation.

The flow would be like so

- Partition Reader Thread would query the records, create batches and dispatch them to the worker threads.
- Reader Thread would wait for the worker threads to complete processing the records (through countdown latches), before it continues to poll for the next set of records.
- The Worker Thread would delete the records that are successfully sent, from the database, and ignores the ones that are failed. The failed sends are logged for errors.
- Once the Worker Thread completes the job, it decreases the countdown latch.
- Reader Thread continues with the next set of records, including the failed ones.

4 Configuration

Property	Description
DB Properties	
db.driverClassName	JDBC Driver class for DB access
db.url	JDBC Url
db.username	DB user
db.encryptedPassword	Encrypted DB password
Misc. DBCP2 supported properties.	
Reader Thread	
partition.names	Comma separated list of partition names to be processed by the application instance. Number of reader threads would be equal to the number of partitions configured.
partition.reader.maxRecordsToFetchFromDb	Number of records that the reader thread should fetch from the partition in a single query.
partition.reader.delayBetweenPollsInSecs	Delay between each poll from the DB for a given Partition Reader Thread
Worker Thread	
sender.threads.per.partition	Number of Sender/Worker threads to be assigned per partition thread. Worker thread pool size is automatically created based on this setting.
Kafka	
Kafka Properties	All Kafka specific properties from the current producer application would be retained.

5 Supportability

- Total time taken by the Reader Thread from polling the DB to successful processing of those records would be logged as INFO
- Total time taken to process a batch by the worker thread would be logged as INFO.
- Time taken by DB fetches and deletes would be logged.
- Kafka errors would be logged.
- Kafka send time for each record would be logged as INFO
- Log4j2 would be supported for enabling logs for 3rd party libraries (like Spring or Kafka).

6 QUERY

PARTITION_VAR will differ by thread P1 - P10

numRows - max number of rows to return

```
SELECT * FROM ( SELECT /*+ INDEX_ASC (A KAFKA_STAGE1_TRANSDT_IDX) */
        ROWID,
        OID,
        OID_DATE,
        TRANS_TIMESTAMP,
        MESSAGE
    FROM KAFKA_STAGE PARTITION ( PARTITION_VAR )
    ORDER BY TRANS_TIMESTAMP )
WHERE ROWNUM <= :numRows
```

BD: The index is partition_key, trans_timestamp but i think trans_timestamp alone would be better, also if we add a status column, that would be appended to the index

Example strings that can be used to test, remember to include the 0x1d chars that are between the different service substrings

https://aciww.sharepoint.com/:t:/r/sites/pd/team/merchantfraud/Teams/Ferrous/Development/Design%20Documents/TEMP/kafkaStringExamples.txt?csf=1&e=gcjknO

7 Future Work

- Explore **asynchronous** Kafka sends to reduce the block time of the Reader Threads.
- Convert the application into a microservice deployable to OpenShift.

8 Producer Application Logs

Following are the INFO Logs from the application. The default log level for the application classes is INFO. Default for 3rd party is WARN.

Component	Log Message	LogType	Comment
TransactionProducer	04-22-2020 20:13:12.097 [main] INFO TransactionProducer - Initializing Producer service...	INFO	Logged only once on startup

ApplicationConfiguration	04-22-2020 20:13:14.439 [main] INFO ApplicationConfiguration - Transaction Producer Service Configuration:
	  partition.names = P1,P2,P3,P4,P5,P6,P7,P8,P9,P10
	  partition.reader.maxRecordsToFetchFromDb = 30
	  partition.reader.delayBetweenPollsInSecs = 10
	  sender.threads.per.partition = 2
	  sender.max.records.delete.perquery =
	  set.add_processing_rows =
	  cardno.encryption.key = [hidden]
	  cardno.cipher.password = [hidden]
	  producer.prop = [hidden]
	  db.driverClassName = oracle.jdbc.driver.OracleDriver
	  db.url = jdbc:oracle:thin:@RSDEVNJ3.am.tsacorp.com:1521:DEVNJ3
	  db.username = sdb
	  db.encryptedPassword = [hidden]
	  db.initialSize = 5
	  db.maxTotal = 50
	  db.maxWaitMillis = 10000
	  db.maxIdle = 1
	  db.connectionProperties = WireProtocolMode=2;defaultRowPrefetch=10
	  db.logAbandoned = true
	  db.removeAbandonedOnBorrow = true
	  db.removeAbandonedOnMaintenance = true
	  db.removeAbandonedTimeout = 30
	  kafka.server.bootstrap.servers = cov3lddbsgem04.ose.am.tsacorp.com:9093,cov3lddbsgem05.ose.am.tsacorp.com:9093,cov3lddbsgem06.ose.am.tsacorp.com:9093
	  kafka.server.schema.registry.url = https://cov3lddbsgem03.ose.am.tsacorp.com:8081
	  kafka.security.enabled = true
	  kafka.security.ssl.keystore.location =
		INFO	Logged only once on startup

Component	Log Message	Log Type	Comment

	\cov3lcep14vm.am.tsacorp.com.keystore.jks
	kafka.security.ssl.truststore.location =
	\cov3lcep14vm.am.tsacorp.com.truststore.jks
	kafka.security.keystore.password.encrypted = [hidden]
	kafka.security.truststore.password.encrypted = [hidden]
	kafka.security.key.password.encrypted = [hidden]
	kafka.security.security.protocol = SASL_SSL
	kafka.security.sasl.kerberos.service.name = kafka
	kafka.security.sasl.mechanism = GSSAPI
	kafka.security.sasl.jaas.config = [hidden]
	kafka.producer.topic = pj-executive
	kafka.producer.key.serializer = org.apache.kafka.common.serializatio
	n.StringSerializer
	kafka.producer.value.serializer = io.confluent.kafka.serializers.KafkaA
	vroSerializer
	kafka.producer.acks = 1
	kafka.producer.retries = 1
	kafka.producer.batch.size = 65536
	kafka.producer.linger.ms = 2
	kafka.producer.buffer.memory = 33554432
	kafka.producer.max.block.ms = 5000
	kafka.producer.request.timeout.ms = 1000
	kafka.producer.max.in.flight.requests.per.connection = 5		

KafkaProducerConfiguration	04-22-2020 20:13:14.457 [main] INFO KafkaProducerConfiguration - Kafka security is enabled	INFO	Logged only once on startup, if SSL is enabled for Kafka.
KafkaProducerConfiguration	04-22-2020 20:13:14.457 [main] INFO KafkaProducerConfiguration - Kafka security is disabled	INFO	Logged only once on startup, if SSL is disabled for Kafka.
TransactionProducer	04-22-2020 20:13:15.352 [main] INFO TransactionProducer - Producer Service initialized.	INFO	Logged only once on startup

Component	Log Message	Log Type	Comment
PartitionReaderThread	04-22-2020 20:13:35.914 [PartitionReaderThread-3] INFO PartitionReaderThread - Total records fetched - 18, time taken to fetch - 20573ms	INFO	Logged by each Reader thread after polling the table for records. If the partition has no records, these logs will not be logged.
TransactionParser	04-22-2020 20:13:35.994 [MessageSenderThread-2] INFO TransactionParser - Begin Processing OID 808002000003UAH20200318061057225	INFO	Logged by the Worker Thread for each record.
TransactionParser	04-22-2020 20:13:36.016 [MessageSenderThread-1] INFO TransactionParser - INFO: Transaction parsing complete for OID 808002000003UAI20200318055947122	INFO	Logged by the Worker Thread for each record.
MessageSenderThread	04-22-2020 21:34:25.022 [MessageSenderThread-10] INFO MessageSenderThread - KAFKA SEND, OID: 808002000003UAI20200318060026033, TIME(s): 1.592	INFO	Logged by the Worker Thread for each record, on successful Kafka send.
MessageSenderThread	04-22-2020 21:34:28.114 [MessageSenderThread-4] INFO MessageSenderThread - Total time taken to process the batch of 5 records - 24010.0ms	INFO	Logged by the Worker Thread once per batch.
PartitionReaderThread	04-22-2020 21:34:28.114 [PartitionReaderThread-10] INFO PartitionReaderThread - Time taken to process the batches - 41514ms	INFO	Logged by the Reader Thread once per poll (after all worker threads complete processing).

Following are the error logs:

KafkaStageDao	04-23-2020 13:13:02.074 [PartitionReaderThread-2] ERROR KafkaStageDao - Error while querying KafkaStage records for partition P2 - PreparedStatementCallback; bad SQL grammar [SELECT * FROM (SELECT /*+ INDEX_ASC (A KAFKA_STAGE1_TRANSDT_IDX) */ ROWID, OID, OID_DATE, TRANS_TIMESTAMP, MESSAGE FROM KAFKA_STAGE PARTITION (P2) ORDER BY TRANS_TIMESTAMP) WHERE ROWNUM <= ?]; nested exception is java.sql.SQLSyntaxErrorException: ORA-00942: table or view does not exist	ERROR	Occurs if the KAFKA_STAGE table or the index does not exist in the database. "Error while querying KafkaStage records for partition" would be logged for any error that we get while fetching the records from DB.
KafkaStageDao	04-23-2020 13:13:02.074 [PartitionReaderThread-2] ERROR KafkaStageDao - Error while deleting the record for oids ……..	ERROR	Errors containing this message would occur if there is a DB error while trying to delete the records from the KAFKA_STAGE table
MessageSenderThread	04-23-2020 13:13:02.074 [PartitionReaderThread-2] ERROR KafkaStageDao - ERROR: Kafka send failed for OID - TEST-OID, Exception - ExecutionException, Error - org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after 250 ms.	ERROR	Logged when Kafka send is timed out for a message
MessageSenderThread	04-23-2020 13:13:02.074 [PartitionReaderThread-2] ERROR KafkaStageDao - ERROR: Kafka send failed for OID - TEST-OID, Exception - ExecutionException, Error - org.apache.kafka.common.errors.NetworkException: The server disconnected before a response was received.	ERROR	Logged when there is a network exception while sending the message to Kafka

MessageSenderThread	ERROR: Kafka send failed for OID - TEST-OID, Exception - ExecutionException, Error - org.apache.kafka.common.errors.TopicAuthorizationException: Not authorized to access topics: [JUNIT-PRODUCER-TOPIC]	ERROR	Logged if the topic is not authorized for the provided keytab credentials.
MessageSenderThread	ERROR: Kafka send failed for OID - TEST-OID, Exception - InterruptedException, Error - Simulated InterruptedException	ERROR	Logged when a Thread Interruption occurs while sending a message to Kafka. Usually seen while shutting down the producer, while it is still processing records.
MessageSenderThread	ERROR: Kafka send failed for OID - TEST-OID, Exception - SerializationException, Error - Error serializing Avro message	ERROR	Logged if the serialization of the message fails during the Kafka send.
MessageSenderThread	ERROR: Kafka send failed for OID - TEST-OID, Exception - ExecutionException, Error - org.apache.kafka.common.errors.RecordTooLargeException: The request included a message larger than the max message size the server will accept.	ERROR	Logged if the record is larger than the max message size. Should ideally not be seen, as we are sending avro messages with strict size restrictions.
MessageSenderThread	ERROR: Kafka send failed for OID - TEST-OID, Exception - ExecutionException, Error - org.apache.kafka.common.errors.ClusterAuthorizationException: The producer is not authorized to do idempotent sends	ERROR	Should not be seen, but it is a potential error thrown by the Kafka producer.

Following are debug and trace logs:

MessageSenderThread	04-22-2020 20:13:35.921 [MessageSenderThread-1] DEBUG MessageSenderThread - Processing txn with oid: 808002000003UAI20200318055947122	DEBUG	Logged by the Worker Thread for each record.
KafkaStageDao	04-22-2020 21:34:28.124 [MessageSenderThread-10] DEBUG KafkaStageDao - Deleted row for OIDs : [808002000003UAI20200318060026033, 808002000003UAO20200318060030431] - 2 rows deleted, time for deletion - 2823.0ms	DEBUG	Logged by the Worker Thread once per batch, on successful Kafka send.
MessageSenderThread	04-22-2020 21:34:28.124 [MessageSenderThread-10] DEBUG MessageSenderThread - Total time taken to process the batch of 5 records - 200ms	DEBUG	Logged by the worker thread after all the messages in the batch are processed.
MessageSenderThread	04-22-2020 21:34:28.124 [MessageSenderThread-10] TRACE MessageSenderThread - Sending to kafka - OID: 808002000003UAI20200318060026033	TRACE	Logged by the worker thread before sending a message to Kafka.
MessageSenderThread	04-24-2020 18:56:42.617 [MessageSenderThread-1] DEBUG MessageSenderThread - recordMetadata 04-24-2020 18:56:42.617 [MessageSenderThread-1] DEBUG MessageSenderThread - hasOffset = true 04-24-2020 18:56:42.618 [MessageSenderThread-1] DEBUG MessageSenderThread - offset = 7171 04-24-2020 18:56:42.618 [MessageSenderThread-1] DEBUG MessageSenderThread - hasTimestamp = true 04-24-2020 18:56:42.618 [MessageSenderThread-1] DEBUG MessageSenderThread - serializedKey = -1 04-24-2020 18:56:42.618 [MessageSenderThread-1] DEBUG MessageSenderThread - serializedValueSize = 2433 04-24-2020 18:56:42.618 [MessageSenderThread-1] DEBUG MessageSenderThread - topic = pj-executive 04-24-2020 18:56:42.618 [MessageSenderThread-17] DEBUG MessageSenderThread - topic = pj-executive 04-24-2020 18:56:42.618 [MessageSenderThread-1] DEBUG MessageSenderThread - partition = 0	DEBUG	Logged if the Kafka send for a message is successful.
MessageSenderThread	04-24-2020 18:56:42.618 [MessageSenderThread-1] TRACE MessageSenderThread - Send complete to kafka.	TRACE	Logged if the Kafka sens is successful for a message

PartitionReaderThread	04-24-2020 18:56:42.618 [PartitionReaderThread-1] TRACE PartitionReaderThread - Polling for stage records.	TRACE	Logged by the Reader thread before it starts polling for the data in the partition.
PartitionReaderThread	04-24-2020 18:56:42.618 [PartitionReaderThread-1] DEBUG PartitionReaderThread - Waiting for 10ms before the next poll.	DEBUG	Logged by the Reader Thread when the previous poll has returned no records.
PartitionReaderThread	04-24-2020 18:56:42.618 [PartitionReaderThread-1] TRACE PartitionReaderThread - All Sender Threads completed.	TRACE	Logged when all worker threads have processed the batches and released back to pool.
PartitionReaderThread	04-24-2020 18:56:42.618 [PartitionReaderThread-1] TRACE PartitionReaderThread - Total batches created - 3	TRACE	Logged by the reader thread before dispatching the batches to the worker threads.
PartitionReaderThread	04-24-2020 18:56:42.618 [PartitionReaderThread-1] TRACE PartitionReaderThread - Dispatched all batches to Sender Threads	TRACE	Logged by the reader thread after dispatching the batches to the worker threads.

9 Producer Configuration

This page details the configuration for the new Kafka producer application.

Full set of configurable properties bundled with the application are available here[4] in BitBucket. These are key properties, however properties of external libraries like Kafka producer and DBCP Datasource can be extended further.

9.1 Jasypt

Passwords for DB and Kafka keystores should be encrypted with Jasypt (secret key for Jasypt should be same as *producer.prop* - mentioned below).

9.1.1 How to use Jasypt for encryption?

1. Download Jasypt distribution (jasypt-1.9.3-dist.zip) from https://github.com/jasypt/jasypt/releases
2. Extract the archive into a folder e.g. C:\jasypt-1.9.3
3. Ensure that "java" is in classpath. Or set the JAVA_HOME appropriately.
4. Go to the bin folder (C:\jasypt-1.9.3\bin) and locate the encrypt script (encrypt.bat or encrypt.sh)
5. To encrypt a password, open a command window and run
    a. encrypt.bat input=<The text you want to encrypt> password=<secret password i.e. value of producer.prop>
6. Copy the output value to the properties file.
7. Run the command for each password that needs to be encrypted. In the example below the input parameter is the password used to connect to the DB. The password parameter is the jasypt key used in encrypting the password.

```
C:\jasypt-1.9.3\bin>encrypt.bat input=r3perf password=secretpasswordman
----ENVIRONMENT-----------------
Runtime: Oracle Corporation Java HotSpot(TM) 64-Bit Server VM 25.121-b13

----ARGUMENTS-------------------
input: r3perf
password: secretpasswordman

----OUTPUT----------------------
IynBQPnOqe50oJf5aMzZJQ==
```

9.2 Service Configuration Properties

The following properties help configure how the service can be scaled-up or scaled-out to handled higher throughputs.

4 https://bitbucket.am.tsacorp.com/projects/RED/repos/operationaldatamanagement/browse/producer/src/main/resources/application.properties?at=refs%2Fheads%2Ffeature%2FMERF-22450-NewProducerChanges

Property	**Description**
partition.names	Takes a comma separated list of partitions that the current instance should handle. e.g. partition.names=P1,P2 would process records only from partitions P1 and P2. A separate "Reader Thread" is created for each partition, which fetches the data from the partition. If a single instance is run to process all the partitions of an FE, then this property should be set with all the partitions. Ensure that each instance of the application has a unique set of partitions assigned to it. No partition should be assigned to multiple instances, else it will process the same record multiple times causing duplicates.
partition.reader.maxRecordsToFetchFromDb	Maximum number of records that the partition reader should fetch from DB on a single poll query. These records are split into batches and assigned to worker threads.
sender.threads.per.partition	Number of Sender/Worker threads to be assigned per Reader Thread. This creates the worker thread pool such that a minimum of (numPartitions x numWorkerThreadsPerPartition) are created. e.g. if number of partitions assigned is 2 and this property is set to 3, then a pool with min. of 6 threads is created. The size of the batch of records delegated to each worker thread is calculated as (partition.reader.maxRecordsToFetchFromDb / sender.threads.per.partition). So, if the number of records fetched are 30 and the sender.threads.per.partition is 3, then the batch size is 10.
partition.reader.delayBetweenPollsInSecs	Delay between each poll from the DB for a given Partition Reader Thread. This is in seconds. Set to 0, if no delay is required. This delay would be applied only if the previous poll has returned no records. This helps avoid unnecessary queries to the database, when the load is low.
sender.max.records.delete.perquery (optional)	Worker thread deletes the records in the batch that are successfully sent out to Kafka. It does so by executing the query DELETE FROM KAFKA_STAGE WHERE ROWID IN (<<List of rowids>>) This property defines the maximum number of records to be deleted by a single query. This is an optional property and is defaulted to batch size if not set. Setting this to 1 is equivalent to deleting one record at a time. The property helps in reducing the number of delete queries and also protect it from *ORA-01795* error by setting an upper limit.

The following properties are used for encryption/descryption of keys and data.

Property	Description
producer.prop	This is the password for the Jasypt encryptor, which is used for encrypting and decrypting keys and password e.g. DB password, Kafka store keys, etc. All properties in application.properties with encrypted values are encrypted with this encryptor and the application used the same to decrypt them.
cardno.cipher.password	Cipher password for the Card Encryptor. The Card encryptor is used for encrypting card data before sending it out to Kafka.
cardno.encryption.key	Key for the Card Encryptor. This should be encrypted with the Jasypt encryptor (with password provided in producer.prop).

9.3 DB Properties

The application used apache-commons' DBCP2 BasicDataSource for DB connection pooling. All DB configuration properties are prefixed with "**db.**". The string in the property without the prefix matches the properties in BasicDatasource (application uses ConfigurationProperties[5] to filter the properties). So, when in doubt, refer the documentation for BasicDataSource[6].

Property	Description
Mandatory Properties	
db.driverClassName	JDBC Driver class name. Since, the target DB is Oracle, the property value can be left as `oracle.jdbc.driver.OracleDriver`
db.url	JDBC connection URL. For the OracleDriver, the url should be of the form jdbc:oracle:thin:@<hostname>:<hostport>:<SID>
db.username	DB user name
db.encryptedPassword	DB password. This should be encrypted with the Jasypt encryptor (with password provided in producer.prop)
***Pooling Properties** - These are basic properties that control the connection pool size and behavior. For additional properties refer the documentation for BasicDataSource.*	
db.initialSize	The initial number of connections that are created when the pool is started (default is 5).

5 https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/context/properties/ConfigurationProperties.html
6 http://commons.apache.org/proper/commons-dbcp/api-2.1.1/org/apache/commons/dbcp2/BasicDataSource.html

Property	**Description**
db.maxTotal	The maximum number of active connections that can be allocated from this pool at the same time, or negative for no limit. (default is 100)
db.maxWaitMillis	The maximum number of milliseconds that the pool will wait (when there are no available connections) for a connection to be returned before throwing an exception, or <= 0 to wait indefinitely. (default is 10000)
db.maxIdle	The maximum number of connections that can remain idle in the pool, without extra ones being destroyed, or negative for no limit. (default is 1)
db.connectionProperties	Custom driver specific properties. Set to the following in the released properties. `WireProtocolMode=2;defaultRowPrefetch=10`
db.logAbandoned	Flag to log stack traces for application code which abandoned a Statement or Connection. (default is true)
db.removeAbandonedOnBorrow	Flag to remove abandoned connections, when a connection is borrowed, if they exceed the removeAbandonedTimout. (default is true)
db.removeAbandonedOnMaintenance	Flag to remove abandoned connections, during maintenance, if they exceed the removeAbandonedTimout. (default is true)
db.removeAbandonedTimeout	Timeout in seconds before an abandoned connection can be removed. (default is 30)

9.4 Kafka Properties

The Kafka properties are separated into 3 sets. Server properties for urls, hostnames, etc. Security properties for securing the connections. And Producer properties for configuring the producer component. These properties are prefixed for categorization. The prefixes are removed before the properties are used for creating the KafkaProducer. If additional properties are required to configure the producer, then they can be added by following the prefix convention.

The following are Kafka Server properties. These are prefixed with "kafka.server".

Property	**Description**
kafka.server.bootstrap.servers	A list of comma separated host/port pairs to use for establishing the initial connection to the Kafka cluster.
kafka.server.schema.registry.url	URL for the schema registry

The following properties are used for configuring Kafka connection security. These are prefixed with "kafka.security".

Property	Description
kafka.security.enabled	Flag to enable/disable secure connections to a Kafka broker. Default is true. If this is set to false, then all properties with prefix '*kafka.security*' are ignored. This should not be set to false in any of the common environments, except developer's own environment.
kafka.security.ssl.keystore.location	The location of the key store file.
kafka.security.ssl.truststore.location	The location of the trust store file.
kafka.security.keystore.password.encrypted	Encrypted store password for the key store file. This should be encrypted with the Jasypt encryptor (with password provided in producer.prop)
kafka.security.truststore.password.encrypted	Encrypted password for the trust store file. This should be encrypted with the Jasypt encryptor (with password provided in producer.prop)
kafka.security.key.password.encrypted	Encrypted password of the private key in the key store file. This should be encrypted with the Jasypt encryptor (with password provided in producer.prop)
kafka.security.security.protocol	Authentication protocol for Kafka e.g. SASL_SSL
kafka.security.sasl.kerberos.service.name	The Kerberos principal name that Kafka runs as.
kafka.security.sasl.mechanism	SASL mechanism used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism. e.g. GSSAPI
kafka.security.sasl.jaas.config	JAAS login context parameters for SASL connections in the format used by JAAS configuration files.

The following properties are used for configuring the producer. These are prefixed with "kafka.producer".

Property	Description
kafka.producer.topic	Kafka Topic to which to send the messages
kafka.producer.key.serializer	Serializer for the key. Should be org.apache.kafka.common.serialization.StringSerializer

Property	Description
kafka.producer.value.serializer	Serializer for the value (i.e. ReDShieldTransaction). Should be io.confluent.kafka.serializers.KafkaAvroSerializer
kafka.producer.acks	The number of acknowledgments the producer requires the leader to have received before considering a request complete.
kafka.producer.retries	Number of times the producer will retry a send to the broker.
kafka.producer.batch.size	The producer will attempt to batch records together into fewer requests whenever multiple records are being sent to the same partition. This property controls the batch size.
kafka.producer.linger.ms	Number of milliseconds the producer would wait before sending out the messages to the broker. Helps improve performance, when used in conjunction with batch.size
kafka.producer.buffer.memory	The total bytes of memory the producer can use to buffer records waiting to be sent to the server.
kafka.producer.max.block.ms	The configuration to control blocking of kafka producer sends; the sends can be blocked either because the buffer is full or metadata unavailable.
kafka.producer.request.timeout.ms	The configuration controls the maximum amount of time the client will wait for the response of a request.
kafka.producer.max.in.flight.requests.per.connection	The maximum number of unacknowledged requests the client will send on a single connection before blocking.