Producer Configuration

ReD Merchant Fraud

Exported on 06/05/2020

Table of Contents

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

Full set of configurable properties bundled with the application are available here[1] in BitBucket. These are key properties, however properties of external libraries like Kafka producer and DBCP Datasource can be extended further.

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

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).

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==
```

2 Service Configuration Properties

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

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).

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[2] to filter the properties). So, when in doubt, refer the documentation for BasicDataSource[3].

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).
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`

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

Property	Description
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)

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.

Property	Description
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
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.