package com.twitter.recosinjector.uua\_processors

import com.twitter.conversions.DurationOps.\_

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.finatra.kafka.consumers.FinagleKafkaConsumerBuilder

import com.twitter.finatra.kafka.domain.KafkaGroupId

import com.twitter.finatra.kafka.domain.SeekStrategy

import com.twitter.finatra.kafka.serde.ScalaSerdes

import com.twitter.finatra.kafka.serde.UnKeyed

import com.twitter.finatra.kafka.serde.UnKeyedSerde

import org.apache.kafka.clients.CommonClientConfigs

import org.apache.kafka.common.config.SaslConfigs

import org.apache.kafka.common.config.SslConfigs

import org.apache.kafka.common.security.auth.SecurityProtocol

import com.twitter.unified\_user\_actions.thriftscala.UnifiedUserAction

import com.twitter.kafka.client.processor.AtLeastOnceProcessor

import com.twitter.kafka.client.processor.ThreadSafeKafkaConsumerClient

import com.twitter.conversions.StorageUnitOps.\_

class UnifiedUserActionsConsumer(

processor: UnifiedUserActionProcessor,

truststoreLocation: String

)(

implicit statsReceiver: StatsReceiver) {

import UnifiedUserActionsConsumer.\_

private val kafkaClient = new ThreadSafeKafkaConsumerClient[UnKeyed, UnifiedUserAction](

FinagleKafkaConsumerBuilder[UnKeyed, UnifiedUserAction]()

.groupId(KafkaGroupId(uuaRecosInjectorGroupId))

.keyDeserializer(UnKeyedSerde.deserializer)

.valueDeserializer(ScalaSerdes.Thrift[UnifiedUserAction].deserializer)

.dest(uuaDest)

.maxPollRecords(maxPollRecords)

.maxPollInterval(maxPollInterval)

.fetchMax(fetchMax)

.seekStrategy(SeekStrategy.END)

.enableAutoCommit(false) // AtLeastOnceProcessor performs commits manually

.withConfig(CommonClientConfigs.SECURITY\_PROTOCOL\_CONFIG, SecurityProtocol.SASL\_SSL.toString)

.withConfig(SslConfigs.SSL\_TRUSTSTORE\_LOCATION\_CONFIG, truststoreLocation)

.withConfig(SaslConfigs.SASL\_MECHANISM, SaslConfigs.GSSAPI\_MECHANISM)

.withConfig(SaslConfigs.SASL\_KERBEROS\_SERVICE\_NAME, "kafka")

.withConfig(SaslConfigs.SASL\_KERBEROS\_SERVER\_NAME, "kafka")

.config)

val atLeastOnceProcessor: AtLeastOnceProcessor[UnKeyed, UnifiedUserAction] = {

AtLeastOnceProcessor[UnKeyed, UnifiedUserAction](

name = processorName,

topic = uuaTopic,

consumer = kafkaClient,

processor = processor.apply,

maxPendingRequests = maxPendingRequests,

workerThreads = workerThreads,

commitIntervalMs = commitIntervalMs,

statsReceiver = statsReceiver.scope(processorName)

)

}

}

object UnifiedUserActionsConsumer {

val maxPollRecords = 1000

val maxPollInterval = 5.minutes

val fetchMax = 1.megabytes

val maxPendingRequests = 1000

val workerThreads = 16

val commitIntervalMs = 10.seconds.inMilliseconds

val processorName = "unified\_user\_actions\_processor"

val uuaTopic = "unified\_user\_actions\_engagements"

val uuaDest = "/s/kafka/bluebird-1:kafka-tls"

val uuaRecosInjectorGroupId = "recos-injector"

}