package com.twitter.unified\_user\_actions.service.module

import com.twitter.inject.TwitterModule

import com.twitter.unified\_user\_actions.kafka.ClientConfigs

import com.twitter.unified\_user\_actions.kafka.CompressionTypeFlag

import com.twitter.util.Duration

import com.twitter.util.StorageUnit

import com.twitter.util.logging.Logging

object FlagsModule extends TwitterModule with Logging {

// Twitter

final val cluster = "cluster"

// Required

final val kafkaSourceCluster = ClientConfigs.kafkaBootstrapServerConfig

final val kafkaDestCluster = ClientConfigs.kafkaBootstrapServerRemoteDestConfig

final val kafkaSourceTopic = "kafka.source.topic"

final val kafkaSinkTopics = "kafka.sink.topics"

final val kafkaGroupId = ClientConfigs.kafkaGroupIdConfig

final val kafkaProducerClientId = ClientConfigs.producerClientIdConfig

final val kafkaMaxPendingRequests = ClientConfigs.kafkaMaxPendingRequestsConfig

final val kafkaWorkerThreads = ClientConfigs.kafkaWorkerThreadsConfig

// Optional

/// Authentication

final val enableTrustStore = ClientConfigs.enableTrustStore

final val trustStoreLocation = ClientConfigs.trustStoreLocationConfig

/// Consumer

final val commitInterval = ClientConfigs.kafkaCommitIntervalConfig

final val maxPollRecords = ClientConfigs.consumerMaxPollRecordsConfig

final val maxPollInterval = ClientConfigs.consumerMaxPollIntervalConfig

final val sessionTimeout = ClientConfigs.consumerSessionTimeoutConfig

final val fetchMax = ClientConfigs.consumerFetchMaxConfig

final val fetchMin = ClientConfigs.consumerFetchMinConfig

final val receiveBuffer = ClientConfigs.consumerReceiveBufferSizeConfig

/// Producer

final val batchSize = ClientConfigs.producerBatchSizeConfig

final val linger = ClientConfigs.producerLingerConfig

final val bufferMem = ClientConfigs.producerBufferMemConfig

final val compressionType = ClientConfigs.compressionConfig

final val retries = ClientConfigs.retriesConfig

final val retryBackoff = ClientConfigs.retryBackoffConfig

final val requestTimeout = ClientConfigs.producerRequestTimeoutConfig

// Twitter

flag[String](

name = cluster,

help = "The zone (or DC) that this service runs, used to potentially filter events"

)

// Required

flag[String](

name = kafkaSourceCluster,

help = ClientConfigs.kafkaBootstrapServerHelp

)

flag[String](

name = kafkaDestCluster,

help = ClientConfigs.kafkaBootstrapServerRemoteDestHelp

)

flag[String](

name = kafkaSourceTopic,

help = "Name of the source Kafka topic"

)

flag[Seq[String]](

name = kafkaSinkTopics,

help = "A list of sink Kafka topics, separated by comma (,)"

)

flag[String](

name = kafkaGroupId,

help = ClientConfigs.kafkaGroupIdHelp

)

flag[String](

name = kafkaProducerClientId,

help = ClientConfigs.producerClientIdHelp

)

flag[Int](

name = kafkaMaxPendingRequests,

help = ClientConfigs.kafkaMaxPendingRequestsHelp

)

flag[Int](

name = kafkaWorkerThreads,

help = ClientConfigs.kafkaWorkerThreadsHelp

)

// Optional

/// Authentication

flag[Boolean](

name = enableTrustStore,

default = ClientConfigs.enableTrustStoreDefault,

help = ClientConfigs.enableTrustStoreHelp

)

flag[String](

name = trustStoreLocation,

default = ClientConfigs.trustStoreLocationDefault,

help = ClientConfigs.trustStoreLocationHelp

)

/// Consumer

flag[Duration](

name = commitInterval,

default = ClientConfigs.kafkaCommitIntervalDefault,

help = ClientConfigs.kafkaCommitIntervalHelp

)

flag[Int](

name = maxPollRecords,

default = ClientConfigs.consumerMaxPollRecordsDefault,

help = ClientConfigs.consumerMaxPollRecordsHelp

)

flag[Duration](

name = maxPollInterval,

default = ClientConfigs.consumerMaxPollIntervalDefault,

help = ClientConfigs.consumerMaxPollIntervalHelp

)

flag[Duration](

name = sessionTimeout,

default = ClientConfigs.consumerSessionTimeoutDefault,

help = ClientConfigs.consumerSessionTimeoutHelp

)

flag[StorageUnit](

name = fetchMax,

default = ClientConfigs.consumerFetchMaxDefault,

help = ClientConfigs.consumerFetchMaxHelp

)

flag[StorageUnit](

name = fetchMin,

default = ClientConfigs.consumerFetchMinDefault,

help = ClientConfigs.consumerFetchMinHelp

)

flag[StorageUnit](

name = receiveBuffer,

default = ClientConfigs.consumerReceiveBufferSizeDefault,

help = ClientConfigs.consumerReceiveBufferSizeHelp

)

/// Producer

flag[StorageUnit](

name = batchSize,

default = ClientConfigs.producerBatchSizeDefault,

help = ClientConfigs.producerBatchSizeHelp

)

flag[Duration](

name = linger,

default = ClientConfigs.producerLingerDefault,

help = ClientConfigs.producerLingerHelp

)

flag[StorageUnit](

name = bufferMem,

default = ClientConfigs.producerBufferMemDefault,

help = ClientConfigs.producerBufferMemHelp

)

flag[CompressionTypeFlag](

name = compressionType,

default = ClientConfigs.compressionDefault,

help = ClientConfigs.compressionHelp

)

flag[Int](

name = retries,

default = ClientConfigs.retriesDefault,

help = ClientConfigs.retriesHelp

)

flag[Duration](

name = retryBackoff,

default = ClientConfigs.retryBackoffDefault,

help = ClientConfigs.retryBackoffHelp

)

flag[Duration](

name = requestTimeout,

default = ClientConfigs.producerRequestTimeoutDefault,

help = ClientConfigs.producerRequestTimeoutHelp

)

}