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

import com.twitter.inject.Test

import com.twitter.kafka.client.headers.ATLA

import com.twitter.kafka.client.headers.Implicits.\_

import com.twitter.kafka.client.headers.PDXA

import com.twitter.kafka.client.headers.Zone

import com.twitter.unified\_user\_actions.service.module.ZoneFiltering

import com.twitter.util.mock.Mockito

import org.apache.kafka.clients.consumer.ConsumerRecord

import org.junit.runner.RunWith

import org.scalatestplus.junit.JUnitRunner

import org.scalatest.prop.TableDrivenPropertyChecks

@RunWith(classOf[JUnitRunner])

class ZoneFilteringTest extends Test with Mockito with TableDrivenPropertyChecks {

trait Fixture {

val consumerRecord =

new ConsumerRecord[Array[Byte], Array[Byte]]("topic", 0, 0l, Array(0), Array(0))

}

test("two DCs filter") {

val zones = Table(

"zone",

Some(ATLA),

Some(PDXA),

None

)

forEvery(zones) { localZoneOpt: Option[Zone] =>

forEvery(zones) { headerZoneOpt: Option[Zone] =>

localZoneOpt.foreach { localZone =>

new Fixture {

headerZoneOpt match {

case Some(headerZone) =>

consumerRecord.headers().setZone(headerZone)

if (headerZone == ATLA && localZone == ATLA)

ZoneFiltering.localDCFiltering(consumerRecord, localZone) shouldBe true

else if (headerZone == PDXA && localZone == PDXA)

ZoneFiltering.localDCFiltering(consumerRecord, localZone) shouldBe true

else

ZoneFiltering.localDCFiltering(consumerRecord, localZone) shouldBe false

case \_ =>

ZoneFiltering.localDCFiltering(consumerRecord, localZone) shouldBe true

}

}

}

}

}

}

}