package com.twitter.frigate.pushservice.predicate

import com.twitter.abdecider.GuestRecipient

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.pushservice.model.PushTypes.Target

import com.twitter.frigate.common.predicate.{FatiguePredicate => CommonFatiguePredicate}

import com.twitter.hermit.predicate.NamedPredicate

import com.twitter.conversions.DurationOps.\_

import com.twitter.frigate.common.util.Experiments.LoggedOutRecsHoldback

import com.twitter.hermit.predicate.Predicate

object LoggedOutTargetPredicates {

def targetFatiguePredicate[T <: Target](

)(

implicit statsReceiver: StatsReceiver

): NamedPredicate[T] = {

val name = "logged\_out\_target\_min\_duration\_since\_push"

CommonFatiguePredicate

.magicRecsPushTargetFatiguePredicate(

minInterval = 24.hours,

maxInInterval = 1

).withStats(statsReceiver.scope(name))

.withName(name)

}

def loggedOutRecsHoldbackPredicate[T <: Target](

)(

implicit statsReceiver: StatsReceiver

): NamedPredicate[T] = {

val name = "logged\_out\_recs\_holdback"

val guestIdNotFoundCounter = statsReceiver.scope("logged\_out").counter("guest\_id\_not\_found")

val controlBucketCounter = statsReceiver.scope("logged\_out").counter("holdback\_control")

val allowTrafficCounter = statsReceiver.scope("logged\_out").counter("allow\_traffic")

Predicate.from { target: T =>

val guestId = target.targetGuestId match {

case Some(guest) => guest

case \_ =>

guestIdNotFoundCounter.incr()

throw new IllegalStateException("guest\_id\_not\_found")

}

target.abDecider

.bucket(LoggedOutRecsHoldback.exptName, GuestRecipient(guestId)).map(\_.name) match {

case Some(LoggedOutRecsHoldback.control) =>

controlBucketCounter.incr()

false

case \_ =>

allowTrafficCounter.incr()

true

}

}

}

}