package com.twitter.frigate.pushservice.send\_handler.generator

import com.twitter.frigate.common.base.MagicFanoutCreatorEventCandidate

import com.twitter.frigate.magic\_events.thriftscala.CreatorFanoutType

import com.twitter.frigate.magic\_events.thriftscala.MagicEventsReason

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

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

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

import com.twitter.frigate.thriftscala.CommonRecommendationType

import com.twitter.frigate.thriftscala.FrigateNotification

import com.twitter.util.Future

object MagicFanoutCreatorEventCandidateGenerator extends CandidateGenerator {

override def getCandidate(

targetUser: PushTypes.Target,

notification: FrigateNotification

): Future[PushTypes.RawCandidate] = {

require(

notification.commonRecommendationType == CommonRecommendationType.CreatorSubscriber || notification.commonRecommendationType == CommonRecommendationType.NewCreator,

"MagicFanoutCreatorEvent: unexpected CRT " + notification.commonRecommendationType

)

require(

notification.creatorSubscriptionNotification.isDefined,

"MagicFanoutCreatorEvent: creatorSubscriptionNotification is not defined")

require(

notification.creatorSubscriptionNotification.exists(\_.magicFanoutPushId.isDefined),

"MagicFanoutCreatorEvent: magicFanoutPushId is not defined")

require(

notification.creatorSubscriptionNotification.exists(\_.fanoutReasons.isDefined),

"MagicFanoutCreatorEvent: fanoutReasons is not defined")

require(

notification.creatorSubscriptionNotification.exists(\_.creatorId.isDefined),

"MagicFanoutCreatorEvent: creatorId is not defined")

if (notification.commonRecommendationType == CommonRecommendationType.CreatorSubscriber) {

require(

notification.creatorSubscriptionNotification

.exists(\_.subscriberId.isDefined),

"MagicFanoutCreatorEvent: subscriber id is not defined"

)

}

val creatorSubscriptionNotification = notification.creatorSubscriptionNotification.get

val candidate = new RawCandidate with MagicFanoutCreatorEventCandidate {

override val target: Target = targetUser

override val pushId: Long =

creatorSubscriptionNotification.magicFanoutPushId.get

override val candidateMagicEventsReasons: Seq[MagicEventsReason] =

creatorSubscriptionNotification.fanoutReasons.get

override val creatorFanoutType: CreatorFanoutType =

creatorSubscriptionNotification.creatorFanoutType

override val commonRecType: CommonRecommendationType =

notification.commonRecommendationType

override val frigateNotification: FrigateNotification = notification

override val subscriberId: Option[Long] = creatorSubscriptionNotification.subscriberId

override val creatorId: Long = creatorSubscriptionNotification.creatorId.get

}

Future.value(candidate)

}

}