package com.twitter.unified\_user\_actions.adapter.ads\_callback\_engagements

import com.twitter.ads.eventstream.thriftscala.EngagementEvent

import com.twitter.adserver.thriftscala.EngagementType

import com.twitter.unified\_user\_actions.adapter.ads\_callback\_engagements.AdsCallbackEngagement.\_

object EngagementTypeMappings {

/\*\*

\* Ads could be Tweets or non-Tweets. Since UUA explicitly sets the item type, it is

\* possible that one Ads Callback engagement type maps to multiple UUA action types.

\*/

def getEngagementMappings(

engagementEvent: Option[EngagementEvent]

): Seq[BaseAdsCallbackEngagement] = {

val promotedTweetId: Option[Long] =

engagementEvent.flatMap(\_.impressionData).flatMap(\_.promotedTweetId)

engagementEvent

.map(event =>

event.engagementType match {

case EngagementType.Fav => Seq(PromotedTweetFav)

case EngagementType.Unfav => Seq(PromotedTweetUnfav)

case EngagementType.Reply => Seq(PromotedTweetReply)

case EngagementType.Retweet => Seq(PromotedTweetRetweet)

case EngagementType.Block => Seq(PromotedTweetBlockAuthor)

case EngagementType.Unblock => Seq(PromotedTweetUnblockAuthor)

case EngagementType.Send => Seq(PromotedTweetComposeTweet)

case EngagementType.Detail => Seq(PromotedTweetClick)

case EngagementType.Report => Seq(PromotedTweetReport)

case EngagementType.Follow => Seq(PromotedProfileFollow)

case EngagementType.Unfollow => Seq(PromotedProfileUnfollow)

case EngagementType.Mute => Seq(PromotedTweetMuteAuthor)

case EngagementType.ProfilePic => Seq(PromotedTweetClickProfile)

case EngagementType.ScreenName => Seq(PromotedTweetClickProfile)

case EngagementType.UserName => Seq(PromotedTweetClickProfile)

case EngagementType.Hashtag => Seq(PromotedTweetClickHashtag)

case EngagementType.Url => Seq(PromotedTweetOpenLink)

case EngagementType.CarouselSwipeNext => Seq(PromotedTweetCarouselSwipeNext)

case EngagementType.CarouselSwipePrevious => Seq(PromotedTweetCarouselSwipePrevious)

case EngagementType.DwellShort => Seq(PromotedTweetLingerImpressionShort)

case EngagementType.DwellMedium => Seq(PromotedTweetLingerImpressionMedium)

case EngagementType.DwellLong => Seq(PromotedTweetLingerImpressionLong)

case EngagementType.SpotlightClick => Seq(PromotedTweetClickSpotlight)

case EngagementType.SpotlightView => Seq(PromotedTweetViewSpotlight)

case EngagementType.TrendView => Seq(PromotedTrendView)

case EngagementType.TrendClick => Seq(PromotedTrendClick)

case EngagementType.VideoContentPlayback25 => Seq(PromotedTweetVideoPlayback25)

case EngagementType.VideoContentPlayback50 => Seq(PromotedTweetVideoPlayback50)

case EngagementType.VideoContentPlayback75 => Seq(PromotedTweetVideoPlayback75)

case EngagementType.VideoAdPlayback25 if promotedTweetId.isDefined =>

Seq(PromotedTweetVideoAdPlayback25)

case EngagementType.VideoAdPlayback25 if promotedTweetId.isEmpty =>

Seq(TweetVideoAdPlayback25)

case EngagementType.VideoAdPlayback50 if promotedTweetId.isDefined =>

Seq(PromotedTweetVideoAdPlayback50)

case EngagementType.VideoAdPlayback50 if promotedTweetId.isEmpty =>

Seq(TweetVideoAdPlayback50)

case EngagementType.VideoAdPlayback75 if promotedTweetId.isDefined =>

Seq(PromotedTweetVideoAdPlayback75)

case EngagementType.VideoAdPlayback75 if promotedTweetId.isEmpty =>

Seq(TweetVideoAdPlayback75)

case EngagementType.DismissRepetitive => Seq(PromotedTweetDismissRepetitive)

case EngagementType.DismissSpam => Seq(PromotedTweetDismissSpam)

case EngagementType.DismissUninteresting => Seq(PromotedTweetDismissUninteresting)

case EngagementType.DismissWithoutReason => Seq(PromotedTweetDismissWithoutReason)

case \_ => Nil

}).toSeq.flatten

}

}