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

import com.twitter.ads.spendserver.thriftscala.SpendServerEvent

import com.twitter.unified\_user\_actions.thriftscala.ActionType

import com.twitter.unified\_user\_actions.thriftscala.AuthorInfo

import com.twitter.unified\_user\_actions.thriftscala.TweetVideoWatch

import com.twitter.unified\_user\_actions.thriftscala.Item

import com.twitter.unified\_user\_actions.thriftscala.TweetActionInfo

import com.twitter.unified\_user\_actions.thriftscala.TweetInfo

abstract class BaseVideoAdsCallbackEngagement(actionType: ActionType)

extends BaseAdsCallbackEngagement(actionType = actionType) {

override def getItem(input: SpendServerEvent): Option[Item] = {

input.engagementEvent.flatMap { e =>

e.impressionData.flatMap { i =>

getTweetInfo(i.promotedTweetId, i.organicTweetId, i.advertiserId, input)

}

}

}

private def getTweetInfo(

promotedTweetId: Option[Long],

organicTweetId: Option[Long],

advertiserId: Long,

input: SpendServerEvent

): Option[Item] = {

val actionedTweetIdOpt: Option[Long] =

if (promotedTweetId.isEmpty) organicTweetId else promotedTweetId

actionedTweetIdOpt.map { actionTweetId =>

Item.TweetInfo(

TweetInfo(

actionTweetId = actionTweetId,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(advertiserId))),

tweetActionInfo = Some(

TweetActionInfo.TweetVideoWatch(

TweetVideoWatch(

isMonetizable = Some(true),

videoOwnerId = input.engagementEvent

.flatMap(e => e.cardEngagement).flatMap(\_.amplifyDetails).flatMap(\_.videoOwnerId),

videoUuid = input.engagementEvent

.flatMap(\_.cardEngagement).flatMap(\_.amplifyDetails).flatMap(\_.videoUuid),

prerollOwnerId = input.engagementEvent

.flatMap(e => e.cardEngagement).flatMap(\_.amplifyDetails).flatMap(

\_.prerollOwnerId),

prerollUuid = input.engagementEvent

.flatMap(\_.cardEngagement).flatMap(\_.amplifyDetails).flatMap(\_.prerollUuid)

))

)

),

)

}

}

}