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

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

import com.twitter.unified\_user\_actions.adapter.common.AdapterUtils

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

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

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

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

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

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

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

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

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

abstract class BaseAdsCallbackEngagement(actionType: ActionType) {

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

input.engagementEvent.flatMap { e =>

e.impressionData.flatMap { i =>

getPromotedTweetInfo(i.promotedTweetId, i.advertiserId)

}

}

}

protected def getPromotedTweetInfo(

promotedTweetIdOpt: Option[Long],

advertiserId: Long,

tweetActionInfoOpt: Option[TweetActionInfo] = None

): Option[Item] = {

promotedTweetIdOpt.map { promotedTweetId =>

Item.TweetInfo(

TweetInfo(

actionTweetId = promotedTweetId,

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

tweetActionInfo = tweetActionInfoOpt)

)

}

}

def getUUA(input: SpendServerEvent): Option[UnifiedUserAction] = {

val userIdentifier: UserIdentifier =

UserIdentifier(

userId = input.engagementEvent.flatMap(e => e.clientInfo.flatMap(\_.userId64)),

guestIdMarketing = input.engagementEvent.flatMap(e => e.clientInfo.flatMap(\_.guestId)),

)

getItem(input).map { item =>

UnifiedUserAction(

userIdentifier = userIdentifier,

item = item,

actionType = actionType,

eventMetadata = getEventMetadata(input),

)

}

}

protected def getEventMetadata(input: SpendServerEvent): EventMetadata =

EventMetadata(

sourceTimestampMs = input.engagementEvent

.map { e => e.engagementEpochTimeMilliSec }.getOrElse(AdapterUtils.currentTimestampMs),

receivedTimestampMs = AdapterUtils.currentTimestampMs,

sourceLineage = SourceLineage.ServerAdsCallbackEngagements,

language = input.engagementEvent.flatMap { e => e.clientInfo.flatMap(\_.languageCode) },

countryCode = input.engagementEvent.flatMap { e => e.clientInfo.flatMap(\_.countryCode) },

clientAppId =

input.engagementEvent.flatMap { e => e.clientInfo.flatMap(\_.clientId) }.map { \_.toLong },

)

}