package com.twitter.unified\_user\_actions.adapter.client\_event

import com.twitter.finagle.stats.NullStatsReceiver

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.clientapp.thriftscala.EventNamespace

import com.twitter.clientapp.thriftscala.LogEvent

import com.twitter.finatra.kafka.serde.UnKeyed

import com.twitter.unified\_user\_actions.adapter.AbstractAdapter

import com.twitter.unified\_user\_actions.adapter.client\_event.ClientEventImpression.\_

import com.twitter.unified\_user\_actions.adapter.client\_event.ClientEventEngagement.\_

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

import scala.util.matching.Regex

class ClientEventAdapter extends AbstractAdapter[LogEvent, UnKeyed, UnifiedUserAction] {

import ClientEventAdapter.\_

override def adaptOneToKeyedMany(

input: LogEvent,

statsReceiver: StatsReceiver = NullStatsReceiver

): Seq[(UnKeyed, UnifiedUserAction)] =

adaptEvent(input).map { e => (UnKeyed, e) }

}

object ClientEventAdapter {

// Refer to go/cme-scribing and go/interaction-event-spec for details

def isVideoEvent(element: String): Boolean = Seq[String](

"gif\_player",

"periscope\_player",

"platform\_amplify\_card",

"video\_player",

"vine\_player").contains(element)

/\*\*

\* Tweet clicks on the Notification Tab on iOS are a special case because the `element` is different

\* from Tweet clicks everywhere else on the platform.

\*

\* For Notification Tab on iOS, `element` could be one of `user\_mentioned\_you`,

\* `user\_mentioned\_you\_in\_a\_quote\_tweet`, `user\_replied\_to\_your\_tweet`, or `user\_quoted\_your\_tweet`.

\*

\* In other places, `element` = `tweet`.

\*/

def isTweetClickEvent(element: String): Boolean =

Seq[String](

"tweet",

"user\_mentioned\_you",

"user\_mentioned\_you\_in\_a\_quote\_tweet",

"user\_replied\_to\_your\_tweet",

"user\_quoted\_your\_tweet"

).contains(element)

final val validUASIosClientIds = Seq[Long](

129032L, // Twitter for iPhone

191841L // Twitter for iPad

)

// Twitter for Android

final val validUASAndroidClientIds = Seq[Long](258901L)

def adaptEvent(inputLogEvent: LogEvent): Seq[UnifiedUserAction] =

Option(inputLogEvent).toSeq

.filterNot { logEvent: LogEvent =>

shouldIgnoreClientEvent(logEvent.eventNamespace)

}

.flatMap { logEvent: LogEvent =>

val actionTypesPerEvent: Seq[BaseClientEvent] = logEvent.eventNamespace.toSeq.flatMap {

name =>

(name.page, name.section, name.component, name.element, name.action) match {

case (\_, \_, \_, \_, Some("favorite")) => Seq(TweetFav)

case (\_, \_, \_, \_, Some("unfavorite")) => Seq(TweetUnfav)

case (\_, \_, Some("stream"), Some("linger"), Some("results")) =>

Seq(TweetLingerImpression)

case (\_, \_, Some("stream"), None, Some("results")) =>

Seq(TweetRenderImpression)

case (\_, \_, \_, \_, Some("send\_reply")) => Seq(TweetReply)

// Different clients may have different actions of the same "send quote"

// but it turns out that both send\_quote and retweet\_with\_comment should correspond to

// "send quote"

case (\_, \_, \_, \_, Some("send\_quote\_tweet")) |

(\_, \_, \_, \_, Some("retweet\_with\_comment")) =>

Seq(TweetQuote)

case (\_, \_, \_, \_, Some("retweet")) => Seq(TweetRetweet)

case (\_, \_, \_, \_, Some("unretweet")) => Seq(TweetUnretweet)

case (\_, \_, \_, \_, Some("reply")) => Seq(TweetClickReply)

case (\_, \_, \_, \_, Some("quote")) => Seq(TweetClickQuote)

case (\_, \_, \_, Some(element), Some("playback\_start")) if isVideoEvent(element) =>

Seq(TweetVideoPlaybackStart)

case (\_, \_, \_, Some(element), Some("playback\_complete")) if isVideoEvent(element) =>

Seq(TweetVideoPlaybackComplete)

case (\_, \_, \_, Some(element), Some("playback\_25")) if isVideoEvent(element) =>

Seq(TweetVideoPlayback25)

case (\_, \_, \_, Some(element), Some("playback\_50")) if isVideoEvent(element) =>

Seq(TweetVideoPlayback50)

case (\_, \_, \_, Some(element), Some("playback\_75")) if isVideoEvent(element) =>

Seq(TweetVideoPlayback75)

case (\_, \_, \_, Some(element), Some("playback\_95")) if isVideoEvent(element) =>

Seq(TweetVideoPlayback95)

case (\_, \_, \_, Some(element), Some("play\_from\_tap")) if isVideoEvent(element) =>

Seq(TweetVideoPlayFromTap)

case (\_, \_, \_, Some(element), Some("video\_quality\_view")) if isVideoEvent(element) =>

Seq(TweetVideoQualityView)

case (\_, \_, \_, Some(element), Some("video\_view")) if isVideoEvent(element) =>

Seq(TweetVideoView)

case (\_, \_, \_, Some(element), Some("video\_mrc\_view")) if isVideoEvent(element) =>

Seq(TweetVideoMrcView)

case (\_, \_, \_, Some(element), Some("view\_threshold")) if isVideoEvent(element) =>

Seq(TweetVideoViewThreshold)

case (\_, \_, \_, Some(element), Some("cta\_url\_click")) if isVideoEvent(element) =>

Seq(TweetVideoCtaUrlClick)

case (\_, \_, \_, Some(element), Some("cta\_watch\_click")) if isVideoEvent(element) =>

Seq(TweetVideoCtaWatchClick)

case (\_, \_, \_, Some("platform\_photo\_card"), Some("click")) => Seq(TweetPhotoExpand)

case (\_, \_, \_, Some("platform\_card"), Some("click")) => Seq(CardClick)

case (\_, \_, \_, \_, Some("open\_app")) => Seq(CardOpenApp)

case (\_, \_, \_, \_, Some("install\_app")) => Seq(CardAppInstallAttempt)

case (\_, \_, \_, Some("platform\_card"), Some("vote")) |

(\_, \_, \_, Some("platform\_forward\_card"), Some("vote")) =>

Seq(PollCardVote)

case (\_, \_, \_, Some("mention"), Some("click")) |

(\_, \_, \_, \_, Some("mention\_click")) =>

Seq(TweetClickMentionScreenName)

case (\_, \_, \_, Some(element), Some("click")) if isTweetClickEvent(element) =>

Seq(TweetClick)

case // Follow from the Topic page (or so-called landing page)

(\_, \_, \_, Some("topic"), Some("follow")) |

// Actually not sure how this is generated ... but saw quite some events in BQ

(\_, \_, \_, Some("social\_proof"), Some("follow")) |

// Click on Tweet's caret menu of "Follow (the topic)", it needs to be:

// 1) user follows the Topic already, 2) and clicked on the "Unfollow Topic" first.

(\_, \_, \_, Some("feedback\_follow\_topic"), Some("click")) =>

Seq(TopicFollow)

case (\_, \_, \_, Some("topic"), Some("unfollow")) |

(\_, \_, \_, Some("social\_proof"), Some("unfollow")) |

(\_, \_, \_, Some("feedback\_unfollow\_topic"), Some("click")) =>

Seq(TopicUnfollow)

case (\_, \_, \_, Some("topic"), Some("not\_interested")) |

(\_, \_, \_, Some("feedback\_not\_interested\_in\_topic"), Some("click")) =>

Seq(TopicNotInterestedIn)

case (\_, \_, \_, Some("topic"), Some("un\_not\_interested")) |

(\_, \_, \_, Some("feedback\_not\_interested\_in\_topic"), Some("undo")) =>

Seq(TopicUndoNotInterestedIn)

case (\_, \_, \_, Some("feedback\_givefeedback"), Some("click")) =>

Seq(TweetNotHelpful)

case (\_, \_, \_, Some("feedback\_givefeedback"), Some("undo")) =>

Seq(TweetUndoNotHelpful)

case (\_, \_, \_, Some("report\_tweet"), Some("click")) |

(\_, \_, \_, Some("report\_tweet"), Some("done")) =>

Seq(TweetReport)

case (\_, \_, \_, Some("feedback\_dontlike"), Some("click")) =>

Seq(TweetNotInterestedIn)

case (\_, \_, \_, Some("feedback\_dontlike"), Some("undo")) =>

Seq(TweetUndoNotInterestedIn)

case (\_, \_, \_, Some("feedback\_notabouttopic"), Some("click")) =>

Seq(TweetNotAboutTopic)

case (\_, \_, \_, Some("feedback\_notabouttopic"), Some("undo")) =>

Seq(TweetUndoNotAboutTopic)

case (\_, \_, \_, Some("feedback\_notrecent"), Some("click")) =>

Seq(TweetNotRecent)

case (\_, \_, \_, Some("feedback\_notrecent"), Some("undo")) =>

Seq(TweetUndoNotRecent)

case (\_, \_, \_, Some("feedback\_seefewer"), Some("click")) =>

Seq(TweetSeeFewer)

case (\_, \_, \_, Some("feedback\_seefewer"), Some("undo")) =>

Seq(TweetUndoSeeFewer)

// Only when action = "submit" we get all fields in ReportDetails, such as reportType

// See https://confluence.twitter.biz/pages/viewpage.action?spaceKey=HEALTH&title=Understanding+ReportDetails

case (Some(page), \_, \_, Some("ticket"), Some("submit"))

if page.startsWith("report\_") =>

Seq(TweetReportServer)

case (Some("profile"), \_, \_, \_, Some("block")) =>

Seq(ProfileBlock)

case (Some("profile"), \_, \_, \_, Some("unblock")) =>

Seq(ProfileUnblock)

case (Some("profile"), \_, \_, \_, Some("mute\_user")) =>

Seq(ProfileMute)

case (Some("profile"), \_, \_, \_, Some("report")) =>

Seq(ProfileReport)

case (Some("profile"), \_, \_, \_, Some("show")) =>

Seq(ProfileShow)

case (\_, \_, \_, Some("follow"), Some("click")) => Seq(TweetFollowAuthor)

case (\_, \_, \_, \_, Some("follow")) => Seq(TweetFollowAuthor, ProfileFollow)

case (\_, \_, \_, Some("unfollow"), Some("click")) => Seq(TweetUnfollowAuthor)

case (\_, \_, \_, \_, Some("unfollow")) => Seq(TweetUnfollowAuthor)

case (\_, \_, \_, Some("block"), Some("click")) => Seq(TweetBlockAuthor)

case (\_, \_, \_, Some("unblock"), Some("click")) => Seq(TweetUnblockAuthor)

case (\_, \_, \_, Some("mute"), Some("click")) => Seq(TweetMuteAuthor)

case (\_, \_, \_, Some(element), Some("click")) if isTweetClickEvent(element) =>

Seq(TweetClick)

case (\_, \_, \_, \_, Some("profile\_click")) => Seq(TweetClickProfile, ProfileClick)

case (\_, \_, \_, \_, Some("share\_menu\_click")) => Seq(TweetClickShare)

case (\_, \_, \_, \_, Some("copy\_link")) => Seq(TweetShareViaCopyLink)

case (\_, \_, \_, \_, Some("share\_via\_dm")) => Seq(TweetClickSendViaDirectMessage)

case (\_, \_, \_, \_, Some("bookmark")) => Seq(TweetShareViaBookmark, TweetBookmark)

case (\_, \_, \_, \_, Some("unbookmark")) => Seq(TweetUnbookmark)

case (\_, \_, \_, \_, Some("hashtag\_click")) |

// This scribe is triggered on mobile platforms (android/iphone) when user click on hashtag in a tweet.

(\_, \_, \_, Some("hashtag"), Some("search")) =>

Seq(TweetClickHashtag)

case (\_, \_, \_, \_, Some("open\_link")) => Seq(TweetOpenLink)

case (\_, \_, \_, \_, Some("take\_screenshot")) => Seq(TweetTakeScreenshot)

case (\_, \_, \_, Some("feedback\_notrelevant"), Some("click")) =>

Seq(TweetNotRelevant)

case (\_, \_, \_, Some("feedback\_notrelevant"), Some("undo")) =>

Seq(TweetUndoNotRelevant)

case (\_, \_, \_, \_, Some("follow\_attempt")) => Seq(ProfileFollowAttempt)

case (\_, \_, \_, \_, Some("favorite\_attempt")) => Seq(TweetFavoriteAttempt)

case (\_, \_, \_, \_, Some("retweet\_attempt")) => Seq(TweetRetweetAttempt)

case (\_, \_, \_, \_, Some("reply\_attempt")) => Seq(TweetReplyAttempt)

case (\_, \_, \_, \_, Some("login")) => Seq(CTALoginClick)

case (Some("login"), \_, \_, \_, Some("show")) => Seq(CTALoginStart)

case (Some("login"), \_, \_, \_, Some("success")) => Seq(CTALoginSuccess)

case (\_, \_, \_, \_, Some("signup")) => Seq(CTASignupClick)

case (Some("signup"), \_, \_, \_, Some("success")) => Seq(CTASignupSuccess)

case // Android app running in the background

(Some("notification"), Some("status\_bar"), None, \_, Some("background\_open")) |

// Android app running in the foreground

(Some("notification"), Some("status\_bar"), None, \_, Some("open")) |

// iOS app running in the background

(Some("notification"), Some("notification\_center"), None, \_, Some("open")) |

// iOS app running in the foreground

(None, Some("toasts"), Some("social"), Some("favorite"), Some("open")) |

// m5

(Some("app"), Some("push"), \_, \_, Some("open")) =>

Seq(NotificationOpen)

case (Some("ntab"), Some("all"), Some("urt"), \_, Some("navigate")) =>

Seq(NotificationClick)

case (Some("ntab"), Some("all"), Some("urt"), \_, Some("see\_less\_often")) =>

Seq(NotificationSeeLessOften)

case (Some("notification"), Some("status\_bar"), None, \_, Some("background\_dismiss")) |

(Some("notification"), Some("status\_bar"), None, \_, Some("dismiss")) | (

Some("notification"),

Some("notification\_center"),

None,

\_,

Some("dismiss")

) =>

Seq(NotificationDismiss)

case (\_, \_, \_, Some("typeahead"), Some("click")) => Seq(TypeaheadClick)

case (Some("search"), \_, Some(component), \_, Some("click"))

if component == "relevance\_prompt\_module" || component == "did\_you\_find\_it\_module" =>

Seq(FeedbackPromptSubmit)

case (Some("app"), Some("enter\_background"), \_, \_, Some("become\_inactive"))

if logEvent.logBase

.flatMap(\_.clientAppId)

.exists(validUASIosClientIds.contains(\_)) =>

Seq(AppExit)

case (Some("app"), \_, \_, \_, Some("become\_inactive"))

if logEvent.logBase

.flatMap(\_.clientAppId)

.exists(validUASAndroidClientIds.contains(\_)) =>

Seq(AppExit)

case (\_, \_, Some("gallery"), Some("photo"), Some("impression")) =>

Seq(TweetGalleryImpression)

case (\_, \_, \_, \_, \_)

if TweetDetailsImpression.isTweetDetailsImpression(logEvent.eventNamespace) =>

Seq(TweetDetailsImpression)

case \_ => Nil

}

}

actionTypesPerEvent.map(\_.toUnifiedUserAction(logEvent))

}.flatten

def shouldIgnoreClientEvent(eventNamespace: Option[EventNamespace]): Boolean =

eventNamespace.exists { name =>

(name.page, name.section, name.component, name.element, name.action) match {

case (Some("ddg"), \_, \_, \_, Some("experiment")) => true

case (Some("qig\_ranker"), \_, \_, \_, \_) => true

case (Some("timelinemixer"), \_, \_, \_, \_) => true

case (Some("timelineservice"), \_, \_, \_, \_) => true

case (Some("tweetconvosvc"), \_, \_, \_, \_) => true

case \_ => false

}

}

}