package com.twitter.unified\_user\_actions.adapter.user\_modification\_event

import com.twitter.gizmoduck.thriftscala.UserModification

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

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

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

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

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

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

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

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

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

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

abstract class BaseUserModificationEvent(actionType: ActionType) {

def getUUA(input: UserModification): UnifiedUserAction = {

val userIdentifier: UserIdentifier = UserIdentifier(userId = input.userId)

UnifiedUserAction(

userIdentifier = userIdentifier,

item = getItem(input),

actionType = actionType,

eventMetadata = getEventMetadata(input),

)

}

protected def getItem(input: UserModification): Item =

Item.ProfileInfo(

ProfileInfo(

actionProfileId = input.userId

.getOrElse(throw new IllegalArgumentException("target user\_id is missing"))

)

)

protected def getEventMetadata(input: UserModification): EventMetadata =

EventMetadata(

sourceTimestampMs = input.updatedAtMsec

.getOrElse(throw new IllegalArgumentException("timestamp is required")),

receivedTimestampMs = AdapterUtils.currentTimestampMs,

sourceLineage = SourceLineage.ServerGizmoduckUserModificationEvents,

)

}

/\*\*

\* When there is a new user creation event in Gizmoduck

\*/

object UserCreate extends BaseUserModificationEvent(ActionType.ServerUserCreate) {

override protected def getItem(input: UserModification): Item =

Item.ProfileInfo(

ProfileInfo(

actionProfileId = input.create

.map { user =>

user.id

}.getOrElse(throw new IllegalArgumentException("target user\_id is missing")),

name = input.create.flatMap { user =>

user.profile.map(\_.name)

},

handle = input.create.flatMap { user =>

user.profile.map(\_.screenName)

},

description = input.create.flatMap { user =>

user.profile.map(\_.description)

}

)

)

override protected def getEventMetadata(input: UserModification): EventMetadata =

EventMetadata(

sourceTimestampMs = input.create

.map { user =>

user.updatedAtMsec

}.getOrElse(throw new IllegalArgumentException("timestamp is required")),

receivedTimestampMs = AdapterUtils.currentTimestampMs,

sourceLineage = SourceLineage.ServerGizmoduckUserModificationEvents,

)

}

object UserUpdate extends BaseUserModificationEvent(ActionType.ServerUserUpdate) {

override protected def getItem(input: UserModification): Item =

Item.ProfileInfo(

ProfileInfo(

actionProfileId =

input.userId.getOrElse(throw new IllegalArgumentException("userId is required")),

profileActionInfo = Some(

ProfileActionInfo.ServerUserUpdate(

ServerUserUpdate(updates = input.update.getOrElse(Nil), success = input.success)))

)

)

override protected def getEventMetadata(input: UserModification): EventMetadata =

EventMetadata(

sourceTimestampMs = input.updatedAtMsec.getOrElse(AdapterUtils.currentTimestampMs),

receivedTimestampMs = AdapterUtils.currentTimestampMs,

sourceLineage = SourceLineage.ServerGizmoduckUserModificationEvents,

)

}