package com.twitter.frigate.pushservice.util

import com.twitter.frigate.common.store.deviceinfo.DeviceInfo

import com.twitter.onboarding.task.service.models.external.PermissionState

import com.twitter.permissions\_storage.thriftscala.AppPermission

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Future

object PushAppPermissionUtil {

final val AddressBookPermissionKey = "addressBook"

final val SyncStateKey = "syncState"

final val SyncStateOnValue = "on"

/\*\*

\* Obtains the specified target's App Permissions, based on their primary device.

\* @param targetId Target's Identifier

\* @param permissionName The permission type we are querying for (address book, geolocation, etc.)

\* @param deviceInfoFut Device info of the Target, presented as a Future

\* @param appPermissionStore Readable Store which allows us to query the App Permission Strato Column

\* @return Returns the AppPermission of the Target, presented as a Future

\*/

def getAppPermission(

targetId: Long,

permissionName: String,

deviceInfoFut: Future[Option[DeviceInfo]],

appPermissionStore: ReadableStore[(Long, (String, String)), AppPermission]

): Future[Option[AppPermission]] = {

deviceInfoFut.flatMap { deviceInfoOpt =>

val primaryDeviceIdOpt = deviceInfoOpt.flatMap(\_.primaryDeviceId)

primaryDeviceIdOpt match {

case Some(primaryDeviceId) =>

val queryKey = (targetId, (primaryDeviceId, permissionName))

appPermissionStore.get(queryKey)

case \_ => Future.None

}

}

}

def hasTargetUploadedAddressBook(

appPermissionOpt: Option[AppPermission]

): Boolean = {

appPermissionOpt.exists { appPermission =>

val syncState = appPermission.metadata.get(SyncStateKey)

appPermission.systemPermissionState == PermissionState.On && syncState

.exists(\_.equalsIgnoreCase(SyncStateOnValue))

}

}

}