package com.twitter.tweetypie

package store

import com.twitter.guano.{thriftscala => guano}

import com.twitter.servo.util.Scribe

import com.twitter.takedown.util.TakedownReasons

import com.twitter.tseng.withholding.thriftscala.TakedownReason

import com.twitter.tweetypie.thriftscala.AuditDeleteTweet

object Guano {

case class MalwareAttempt(

url: String,

userId: UserId,

clientAppId: Option[Long],

remoteHost: Option[String]) {

def toScribeMessage: guano.ScribeMessage =

guano.ScribeMessage(

`type` = guano.ScribeType.MalwareAttempt,

malwareAttempt = Some(

guano.MalwareAttempt(

timestamp = Time.now.inSeconds,

host = remoteHost,

userId = userId,

url = url,

`type` = guano.MalwareAttemptType.Status,

clientAppId = clientAppId.map(\_.toInt) // yikes!

)

)

)

}

case class DestroyTweet(

tweet: Tweet,

userId: UserId,

byUserId: UserId,

passthrough: Option[AuditDeleteTweet]) {

def toScribeMessage: guano.ScribeMessage =

guano.ScribeMessage(

`type` = guano.ScribeType.DestroyStatus,

destroyStatus = Some(

guano.DestroyStatus(

`type` = Some(guano.DestroyStatusType.Status),

timestamp = Time.now.inSeconds,

userId = userId,

byUserId = byUserId,

statusId = tweet.id,

text = "",

reason = passthrough

.flatMap(\_.reason)

.flatMap { r => guano.UserActionReason.valueOf(r.name) }

.orElse(Some(guano.UserActionReason.Other)),

done = passthrough.flatMap(\_.done).orElse(Some(true)),

host = passthrough.flatMap(\_.host),

bulkId = passthrough.flatMap(\_.bulkId),

note = passthrough.flatMap(\_.note),

runId = passthrough.flatMap(\_.runId),

clientApplicationId = passthrough.flatMap(\_.clientApplicationId),

userAgent = passthrough.flatMap(\_.userAgent)

)

)

)

}

case class Takedown(

tweetId: TweetId,

userId: UserId,

reason: TakedownReason,

takendown: Boolean,

note: Option[String],

host: Option[String],

byUserId: Option[UserId]) {

def toScribeMessage: guano.ScribeMessage =

guano.ScribeMessage(

`type` = guano.ScribeType.PctdAction,

pctdAction = Some(

guano.PctdAction(

`type` = guano.PctdActionType.Status,

timestamp = Time.now.inSeconds,

tweetId = Some(tweetId),

userId = userId,

countryCode =

TakedownReasons.reasonToCountryCode.applyOrElse(reason, (\_: TakedownReason) => ""),

takendown = takendown,

note = note,

host = host,

byUserId = byUserId.getOrElse(-1L),

reason = Some(reason)

)

)

)

}

case class UpdatePossiblySensitiveTweet(

tweetId: TweetId,

userId: UserId,

byUserId: UserId,

action: guano.NsfwTweetActionAction,

enabled: Boolean,

host: Option[String],

note: Option[String]) {

def toScribeMessage: guano.ScribeMessage =

guano.ScribeMessage(

`type` = guano.ScribeType.NsfwTweetAction,

nsfwTweetAction = Some(

guano.NsfwTweetAction(

timestamp = Time.now.inSeconds,

host = host,

userId = userId,

byUserId = byUserId,

action = action,

enabled = enabled,

note = note,

tweetId = tweetId

)

)

)

}

def apply(

scribe: FutureEffect[guano.ScribeMessage] = Scribe(guano.ScribeMessage,

Scribe("trust\_eng\_audit"))

): Guano = {

new Guano {

override val scribeMalwareAttempt: FutureEffect[MalwareAttempt] =

scribe.contramap[MalwareAttempt](\_.toScribeMessage)

override val scribeDestroyTweet: FutureEffect[DestroyTweet] =

scribe.contramap[DestroyTweet](\_.toScribeMessage)

override val scribeTakedown: FutureEffect[Takedown] =

scribe.contramap[Takedown](\_.toScribeMessage)

override val scribeUpdatePossiblySensitiveTweet: FutureEffect[UpdatePossiblySensitiveTweet] =

scribe.contramap[UpdatePossiblySensitiveTweet](\_.toScribeMessage)

}

}

}

trait Guano {

val scribeMalwareAttempt: FutureEffect[Guano.MalwareAttempt]

val scribeDestroyTweet: FutureEffect[Guano.DestroyTweet]

val scribeTakedown: FutureEffect[Guano.Takedown]

val scribeUpdatePossiblySensitiveTweet: FutureEffect[Guano.UpdatePossiblySensitiveTweet]

}