package com.twitter.tweetypie.additionalfields

import com.twitter.tweetypie.thriftscala.Tweet

import com.twitter.scrooge.TFieldBlob

import com.twitter.scrooge.ThriftStructField

object AdditionalFields {

type FieldId = Short

/\*\* additional fields really start at 100, be we are ignoring conversation id for now \*/

val StartAdditionalId = 101

/\*\* all known [[Tweet]] field IDs \*/

val CompiledFieldIds: Seq[FieldId] = Tweet.metaData.fields.map(\_.id)

/\*\* all known [[Tweet]] fields in the "additional-field" range (excludes id) \*/

val CompiledAdditionalFieldMetaDatas: Seq[ThriftStructField[Tweet]] =

Tweet.metaData.fields.filter(f => isAdditionalFieldId(f.id))

val CompiledAdditionalFieldsMap: Map[Short, ThriftStructField[Tweet]] =

CompiledAdditionalFieldMetaDatas.map(field => (field.id, field)).toMap

/\*\* all known [[Tweet]] field IDs in the "additional-field" range \*/

val CompiledAdditionalFieldIds: Seq[FieldId] =

CompiledAdditionalFieldsMap.keys.toSeq

/\*\* all [[Tweet]] field IDs which should be rejected when set as additional

\* fields on via PostTweetRequest.additionalFields or RetweetRequest.additionalFields \*/

val RejectedFieldIds: Seq[FieldId] = Seq(

// Should be provided via PostTweetRequest.conversationControl field. go/convocontrolsbackend

Tweet.ConversationControlField.id,

// This field should only be set based on whether the client sets the right community

// tweet annotation.

Tweet.CommunitiesField.id,

// This field should not be set by clients and should opt for

// [[PostTweetRequest.ExclusiveTweetControlOptions]].

// The exclusiveTweetControl field requires the userId to be set

// and we shouldn't trust the client to provide the right one.

Tweet.ExclusiveTweetControlField.id,

// This field should not be set by clients and should opt for

// [[PostTweetRequest.TrustedFriendsControlOptions]].

// The trustedFriendsControl field requires the trustedFriendsListId to be

// set and we shouldn't trust the client to provide the right one.

Tweet.TrustedFriendsControlField.id,

// This field should not be set by clients and should opt for

// [[PostTweetRequest.CollabControlOptions]].

// The collabControl field requires a list of Collaborators to be

// set and we shouldn't trust the client to provide the right one.

Tweet.CollabControlField.id

)

def isAdditionalFieldId(fieldId: FieldId): Boolean =

fieldId >= StartAdditionalId

/\*\*

\* Provides a list of all additional field IDs on the tweet, which include all

\* the compiled additional fields and all the provided passthrough fields. This includes

\* compiled additional fields where the value is None.

\*/

def allAdditionalFieldIds(tweet: Tweet): Seq[FieldId] =

CompiledAdditionalFieldIds ++ tweet.\_passthroughFields.keys

/\*\*

\* Provides a list of all field IDs that have a value on the tweet which are not known compiled

\* additional fields (excludes [[Tweet.id]]).

\*/

def unsettableAdditionalFieldIds(tweet: Tweet): Seq[FieldId] =

CompiledFieldIds

.filter { id =>

!isAdditionalFieldId(id) && id != Tweet.IdField.id && tweet.getFieldBlob(id).isDefined

} ++

tweet.\_passthroughFields.keys

/\*\*

\* Provides a list of all field IDs that have a value on the tweet which are explicitly disallowed

\* from being set via PostTweetRequest.additionalFields and RetweetRequest.additionalFields

\*/

def rejectedAdditionalFieldIds(tweet: Tweet): Seq[FieldId] =

RejectedFieldIds

.filter { id => tweet.getFieldBlob(id).isDefined }

def unsettableAdditionalFieldIdsErrorMessage(unsettableFieldIds: Seq[FieldId]): String =

s"request may not contain fields: [${unsettableFieldIds.sorted.mkString(", ")}]"

/\*\*

\* Provides a list of all additional field IDs that have a value on the tweet,

\* compiled and passthrough (excludes Tweet.id).

\*/

def nonEmptyAdditionalFieldIds(tweet: Tweet): Seq[FieldId] =

CompiledAdditionalFieldMetaDatas.collect {

case f if f.getValue(tweet) != None => f.id

} ++ tweet.\_passthroughFields.keys

def additionalFields(tweet: Tweet): Seq[TFieldBlob] =

(tweet.getFieldBlobs(CompiledAdditionalFieldIds) ++ tweet.\_passthroughFields).values.toSeq

/\*\*

\* Merge base tweet with additional fields.

\* Non-additional fields in the additional tweet are ignored.

\* @param base: a tweet that contains basic fields

\* @param additional: a tweet object that carries additional fields

\*/

def setAdditionalFields(base: Tweet, additional: Tweet): Tweet =

setAdditionalFields(base, additionalFields(additional))

def setAdditionalFields(base: Tweet, additional: Option[Tweet]): Tweet =

additional.map(setAdditionalFields(base, \_)).getOrElse(base)

def setAdditionalFields(base: Tweet, additional: Traversable[TFieldBlob]): Tweet =

additional.foldLeft(base) { case (t, f) => t.setField(f) }

/\*\*

\* Unsets the specified fields on the given tweet.

\*/

def unsetFields(tweet: Tweet, fieldIds: Iterable[FieldId]): Tweet = {

tweet.unsetFields(fieldIds.toSet)

}

}