namespace java com.twitter.tweetypie.thriftjava

namespace py gen.twitter.tweetypie.tweet\_events

#@namespace scala com.twitter.tweetypie.thriftscala

#@namespace strato com.twitter.tweetypie

namespace rb TweetyPie

namespace go tweetypie

include "com/twitter/tseng/withholding/withholding.thrift"

include "com/twitter/tweetypie/transient\_context.thrift"

include "com/twitter/tweetypie/tweet.thrift"

include "com/twitter/tweetypie/tweet\_audit.thrift"

include "com/twitter/gizmoduck/user.thrift"

/\*\*

\* SafetyType encodes the event user's safety state in an enum so downstream

\* event processors can filter events without having to load the user.

\*/

enum SafetyType {

PRIVATE = 0 // user.safety.isProtected

RESTRICTED = 1 // !PRIVATE && user.safety.suspended

PUBLIC = 2 // !(PRIVATE || RESTRICTED)

RESERVED0 = 3

RESERVED1 = 4

RESERVED2 = 5

RESERVED3 = 6

}

struct TweetCreateEvent {

/\*\*

\* The tweet that has been created.

\*/

1: tweet.Tweet tweet

/\*\*

\* The user who owns the created tweet.

\*/

2: user.User user

/\*\*

\* The tweet being retweeted.

\*/

3: optional tweet.Tweet source\_tweet

/\*\*

\* The user who owns source\_tweet.

\*/

4: optional user.User source\_user

/\*\*

\* The user whose tweet or retweet is being retweeted.

\*

\* This is the id of the user who owns

\* tweet.core\_data.share.parent\_status\_id. In many cases this will be the

\* same as source\_user.id; it is different when the tweet is created via

\* another retweet. See the explanation of source\_user\_id and parent\_user\_id

\* in Share for examples.

\*/

5: optional i64 retweet\_parent\_user\_id (personalDataType = 'UserId')

/\*\*

\* The tweet quoted in the created tweet.

\*/

6: optional tweet.Tweet quoted\_tweet

/\*\*

\* The user who owns quoted\_tweet.

\*/

7: optional user.User quoted\_user

/\*\*

\* Arbitrary passthrough metadata about tweet creation.

\*

\* See TweetCreateContextKey for more details about the data that may be

\* present here.

\*/

8: optional map<tweet.TweetCreateContextKey, string> additional\_context (personalDataTypeValue='UserId')

/\*\*

\* Additional request arguments passed through to consumers.

\*/

9: optional transient\_context.TransientCreateContext transient\_context

/\*\*

\* Flag exposing if a quoted tweet has been quoted by the user previously.

\*\*/

10: optional bool quoter\_has\_already\_quoted\_tweet

}(persisted='true', hasPersonalData = 'true')

struct TweetDeleteEvent {

/\*\*

\* The tweet being deleted.

\*/

1: tweet.Tweet tweet

/\*\*

\* The user who owns the deleted tweet.

\*/

2: optional user.User user

/\*\*

\* Whether this tweet was deleted as part of user erasure (the process of deleting tweets

\* belonging to deactivated accounts).

\*

\* These deletions occur in high volume spikes and the tweets have already been made invisible

\* externally. You may wish to process them in batches or offline.

\*/

3: optional bool is\_user\_erasure

/\*\*

\* Audit information from the DeleteTweetRequest that caused this deletion.

\*

\* This field is used to track the reason for deletion in non-user-initiated

\* tweet deletions, like Twitter support agents deleting tweets or spam

\* cleanup.

\*/

4: optional tweet\_audit.AuditDeleteTweet audit

/\*\*

\* Id of the user initiating this request.

\* It could be either the owner of the tweet or an admin.

\* It is used for scrubbing.

\*/

5: optional i64 by\_user\_id (personalDataType = 'UserId')

/\*\*

\* Whether this tweet was deleted by an admin user or not

\*

\* It is used for scrubbing.

\*/

6: optional bool is\_admin\_delete

}(persisted='true', hasPersonalData = 'true')

struct TweetUndeleteEvent {

1: tweet.Tweet tweet

2: optional user.User user

3: optional tweet.Tweet source\_tweet

4: optional user.User source\_user

5: optional i64 retweet\_parent\_user\_id (personalDataType = 'UserId')

6: optional tweet.Tweet quoted\_tweet

7: optional user.User quoted\_user

// timestamp of the deletion that this undelete is reversing

8: optional i64 deleted\_at\_msec

}(persisted='true', hasPersonalData = 'true')

/\*\*

\* When a user deletes the location information for their tweets, we send one

\* TweetScrubGeoEvent for every tweet from which the location is removed.

\*

\* Users cause this by selecting "Delete location information" in Settings ->

\* Privacy.

\*/

struct TweetScrubGeoEvent {

1: i64 tweet\_id (personalDataType = 'TweetId')

2: i64 user\_id (personalDataType = 'UserId')

}(persisted='true', hasPersonalData = 'true')

/\*\*

\* When a user deletes the location information for their tweets, we send one

\* UserScrubGeoEvent with the max tweet ID that was scrubbed (in addition to

\* sending multiple TweetScrubGeoEvents as described above).

\*

\* Users cause this by selecting "Delete location information" in Settings ->

\* Privacy. This additional event is sent to maintain backwards compatibility

\* with Hosebird.

\*/

struct UserScrubGeoEvent {

1: i64 user\_id (personalDataType = 'UserId')

2: i64 max\_tweet\_id (personalDataType = 'TweetId')

}(persisted='true', hasPersonalData = 'true')

struct TweetTakedownEvent {

1: i64 tweet\_id (personalDataType = 'TweetId')

2: i64 user\_id (personalDataType = 'UserId')

// This is the complete list of takedown country codes for the tweet,

// including whatever modifications were made to trigger this event.

// @deprecated Prefer takedown\_reasons once TWEETYPIE-4329 deployed

3: list<string> takedown\_country\_codes = []

// This is the complete list of takedown reasons for the tweet,

// including whatever modifications were made to trigger this event.

4: list<withholding.TakedownReason> takedown\_reasons = []

}(persisted='true', hasPersonalData = 'true')

struct AdditionalFieldUpdateEvent {

// Only contains the tweet id and modified or newly added fields on that tweet.

// Unchanged fields and tweet core data are omitted.

1: tweet.Tweet updated\_fields

2: optional i64 user\_id (personalDataType = 'UserId')

}(persisted='true', hasPersonalData = 'true')

struct AdditionalFieldDeleteEvent {

// a map from tweet id to deleted field ids

// Each event will only contain one tweet.

1: map<i64, list<i16>> deleted\_fields (personalDataTypeKey='TweetId')

2: optional i64 user\_id (personalDataType = 'UserId')

}(persisted='true', hasPersonalData = 'true')

// This event is only logged to scribe not sent to EventBus

struct TweetMediaTagEvent {

1: i64 tweet\_id (personalDataType = 'TweetId')

2: i64 user\_id (personalDataType = 'UserId')

3: set<i64> tagged\_user\_ids (personalDataType = 'UserId')

4: optional i64 timestamp\_ms

}(persisted='true', hasPersonalData = 'true')

struct TweetPossiblySensitiveUpdateEvent {

1: i64 tweet\_id (personalDataType = 'TweetId')

2: i64 user\_id (personalDataType = 'UserId')

// The below two fields contain the results of the update.

3: bool nsfw\_admin

4: bool nsfw\_user

}(persisted='true', hasPersonalData = 'true')

struct QuotedTweetDeleteEvent {

1: i64 quoting\_tweet\_id (personalDataType = 'TweetId')

2: i64 quoting\_user\_id (personalDataType = 'UserId')

3: i64 quoted\_tweet\_id (personalDataType = 'TweetId')

4: i64 quoted\_user\_id (personalDataType = 'UserId')

}(persisted='true', hasPersonalData = 'true')

struct QuotedTweetTakedownEvent {

1: i64 quoting\_tweet\_id (personalDataType = 'TweetId')

2: i64 quoting\_user\_id (personalDataType = 'UserId')

3: i64 quoted\_tweet\_id (personalDataType = 'TweetId')

4: i64 quoted\_user\_id (personalDataType = 'UserId')

// This is the complete list of takedown country codes for the tweet,

// including whatever modifications were made to trigger this event.

// @deprecated Prefer takedown\_reasons

5: list<string> takedown\_country\_codes = []

// This is the complete list of takedown reasons for the tweet,

// including whatever modifications were made to trigger this event.

6: list<withholding.TakedownReason> takedown\_reasons = []

}(persisted='true', hasPersonalData = 'true')

union TweetEventData {

1: TweetCreateEvent tweet\_create\_event

2: TweetDeleteEvent tweet\_delete\_event

3: AdditionalFieldUpdateEvent additional\_field\_update\_event

4: AdditionalFieldDeleteEvent additional\_field\_delete\_event

5: TweetUndeleteEvent tweet\_undelete\_event

6: TweetScrubGeoEvent tweet\_scrub\_geo\_event

7: TweetTakedownEvent tweet\_takedown\_event

8: UserScrubGeoEvent user\_scrub\_geo\_event

9: TweetPossiblySensitiveUpdateEvent tweet\_possibly\_sensitive\_update\_event

10: QuotedTweetDeleteEvent quoted\_tweet\_delete\_event

11: QuotedTweetTakedownEvent quoted\_tweet\_takedown\_event

}(persisted='true', hasPersonalData = 'true')

/\*\*

\* @deprecated

\*/

struct Checksum {

1: i32 checksum

}(persisted='true')

struct TweetEventFlags {

/\*\*

\* @deprecated Was dark\_for\_service.

\*/

1: list<string> unused1 = []

2: i64 timestamp\_ms

3: optional SafetyType safety\_type

/\*\*

\* @deprecated Was checksum.

\*/

4: optional Checksum unused4

}(persisted='true')

/\*\*

\* A TweetEvent is a notification published to the tweet\_events stream.

\*/

struct TweetEvent {

1: TweetEventData data

2: TweetEventFlags flags

}(persisted='true', hasPersonalData = 'true')