namespace java com.twitter.tweetypie.thriftjava.graphql

#@namespace scala com.twitter.tweetypie.thriftscala.graphql

#@namespace strato com.twitter.tweetypie.graphql

/\*\*

\* Reasons for defining "prefetch" structs:

\* i) It enables GraphQL prefetch caching

\* ii) All tweet mutation operations are defined to support prefetch caching for API consistency

\* and future flexibility. (Populating the cache with VF results being a potential use case.)

\*/

include "com/twitter/ads/callback/engagement\_request.thrift"

include "com/twitter/strato/graphql/existsAndPrefetch.thrift"

struct UnretweetRequest {

/\*\*

\* Tweet ID of the source tweet being referenced in the unretweet.

\* Note: The retweet\_id isn't being passed here as it will result in a

\* successful response, but won't have any effect. This is due to

\* how Tweetypie's unretweet endpoint works.

\*/

1: required i64 source\_tweet\_id (

strato.json.numbers.type='string',

strato.description='The source tweet to be unretweeted.'

)

2: optional string comparison\_id (

strato.description='Correlates requests originating from REST endpoints and GraphQL endpoints.'

)

} (strato.graphql.typename='UnretweetRequest')

struct UnretweetResponse {

/\*\*

\* The response contains the source tweet's ID being unretweeted.

\* Reasons for this:

\* i) The operation should return a non-void response to retain consistency

\* with other tweet mutation APIs.

\* ii) The response struct should define at least one field due to requirements

\* of the GraphQL infrastructure.

\* iii) This allows the caller to hydrate the source tweet if required and request

\* updated counts on the source tweet if desired. (since this operation decrements

\* the source tweet's retweet count)

\*/

1: optional i64 source\_tweet\_id (

strato.space='Tweet',

strato.graphql.fieldname='source\_tweet',

strato.description='The source tweet that was unretweeted.'

)

} (strato.graphql.typename='UnretweetResponse')

struct UnretweetResponseWithSubqueryPrefetchItems {

1: optional UnretweetResponse data

2: optional existsAndPrefetch.PrefetchedData subqueryPrefetchItems

}

struct CreateRetweetRequest {

1: required i64 tweet\_id (strato.json.numbers.type='string')

// @see com.twitter.tweetypie.thriftscala.PostTweetRequest.nullcast

2: bool nullcast = 0 (

strato.description='Do not deliver this retweet to a user\'s followers. http://go/nullcast'

)

// @see com.twitter.ads.callback.thriftscala.EngagementRequest

3: optional engagement\_request.EngagementRequest engagement\_request (

strato.description='The ad engagement from which this retweet was created.'

)

// @see com.twitter.tweetypie.thriftscala.PostTweetRequest.PostTweetRequest.comparison\_id

4: optional string comparison\_id (

strato.description='Correlates requests originating from REST endpoints and GraphQL endpoints. UUID v4 (random) 36 character string.'

)

} (strato.graphql.typename='CreateRetweetRequest')

struct CreateRetweetResponse {

1: optional i64 retweet\_id (

strato.space='Tweet',

strato.graphql.fieldname='retweet',

strato.description='The created retweet.'

)

} (strato.graphql.typename='CreateRetweetResponse')

struct CreateRetweetResponseWithSubqueryPrefetchItems {

1: optional CreateRetweetResponse data

2: optional existsAndPrefetch.PrefetchedData subqueryPrefetchItems

}

struct TweetReply {

//@see com.twitter.tweetypie.thriftscala.PostTweetRequest.in\_reply\_to\_tweet\_id

1: i64 in\_reply\_to\_tweet\_id (

strato.json.numbers.type='string',

strato.description='The id of the tweet that this tweet is replying to.'

)

//@see com.twitter.tweetypie.thriftscala.PostTweetRequest.exclude\_reply\_user\_ids

2: list<i64> exclude\_reply\_user\_ids = [] (

strato.json.numbers.type='string',

strato.description='Screen names appearing in the mention prefix can be excluded. Because the mention prefix must always include the leading mention to preserve directed-at addressing for the in-reply-to tweet author, attempting to exclude that user id will have no effect. Specifying a user id not in the prefix will be silently ignored.'

)

} (strato.graphql.typename='TweetReply')

struct TweetMediaEntity {

// @see com.twitter.tweetypie.thriftscala.PostTweetRequest.media\_upload\_ids

1: i64 media\_id (

strato.json.numbers.type='string',

strato.description='Media id as obtained from the User Image Service when uploaded.'

)

// @see com.twitter.tweetypie.thriftscala.Tweet.media\_tags

2: list<i64> tagged\_users = [] (

strato.json.numbers.type='string',

strato.description='List of user\_ids to tag in this media entity. Requires Client App Privelege MEDIA\_TAGS. Contributors (http://go/teams) are not supported. Tags are silently dropped when unauthorized.'

)

} (strato.graphql.typename='TweetMediaEntity')

struct TweetMedia {

1: list<TweetMediaEntity> media\_entities = [] (

strato.description='You may include up to 4 photos or 1 animated GIF or 1 video in a Tweet.'

)

/\*\*

\* @deprecated @see com.twitter.tweetypie.thriftscala.PostTweetRequest.possibly\_sensitive for

\* more details on why this field is ignored.

\*/

2: bool possibly\_sensitive = 0 (

strato.description='Mark this tweet as possibly containing objectionable media.'

)

} (strato.graphql.typename='TweetMedia')

//This is similar to the APITweetAnnotation struct except that here all the id fields are required.

struct TweetAnnotation {

1: i64 group\_id (strato.json.numbers.type='string')

2: i64 domain\_id (strato.json.numbers.type='string')

3: i64 entity\_id (strato.json.numbers.type='string')

} (strato.graphql.typename='TweetAnnotation', strato.case.format='preserve')

struct TweetGeoCoordinates {

1: double latitude (strato.description='The latitude of the location this Tweet refers to. The valid range for latitude is -90.0 to +90.0 (North is positive) inclusive.')

2: double longitude (strato.description='The longitude of the location this Tweet refers to. The valid range for longitude is -180.0 to +180.0 (East is positive) inclusive.')

3: bool display\_coordinates = 1 (strato.description='Whether or not make the coordinates public. When false, geo coordinates are persisted with the Tweet but are not shared publicly.')

} (strato.graphql.typename='TweetGeoCoordinates')

struct TweetGeo {

1: optional TweetGeoCoordinates coordinates (

strato.description='The geo coordinates of the location this Tweet refers to.'

)

2: optional string place\_id (

strato.description='A place in the world. See also https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/geo#place'

)

3: optional string geo\_search\_request\_id (

strato.description='See https://confluence.twitter.biz/display/GEO/Passing+the+geo+search+request+ID'

)

} (

strato.graphql.typename='TweetGeo',

strato.description='Tweet geo location metadata. See https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/geo'

)

enum BatchComposeMode {

BATCH\_FIRST = 1 (strato.description='This is the first Tweet in a batch.')

BATCH\_SUBSEQUENT = 2 (strato.description='This is any of the subsequent Tweets in a batch.')

}(

strato.graphql.typename='BatchComposeMode',

strato.description='Indicates whether a Tweet was created using a batch composer, and if so position of a Tweet within the batch. A value of None, indicates that the tweet was not created in a batch. More info: go/batchcompose.'

)

/\*\*

\* Conversation Controls

\* See also:

\* tweet.thrift/Tweet.conversation\_control

\* tweet\_service.thrift/TweetCreateConversationControl

\* tweet\_service.thrift/PostTweetRequest.conversation\_control

\*

\* These types are isomorphic/equivalent to tweet\_service.thrift/TweetCreateConversationControl\* to

\* avoid exposing internal service thrift types.

\*/

enum ConversationControlMode {

BY\_INVITATION = 1 (strato.description='Users that the conversation owner mentions by @screenname in the tweet text are invited.')

COMMUNITY = 2 (strato.description='The conversation owner, invited users, and users who the conversation owner follows can reply.')

} (

strato.graphql.typename='ConversationControlMode'

)

struct TweetConversationControl {

1: ConversationControlMode mode

} (

strato.graphql.typename='TweetConversationControl',

strato.description='Specifies limits on user participation in a conversation. See also http://go/dont-at-me. Up to one value may be provided. (Conceptually this is a union, however graphql doesn\'t support union types as inputs.)'

)

// empty for now, but intended to be populated in later iterations of the super follows project.

struct ExclusiveTweetControlOptions {} (

strato.description='Marks a tweet as exclusive. See go/superfollows.',

strato.graphql.typename='ExclusiveTweetControlOptions',

)

struct EditOptions {

1: optional i64 previous\_tweet\_id (strato.json.numbers.type='string', strato.description='previous Tweet id')

} (

strato.description='Edit options for a Tweet.',

strato.graphql.typename='EditOptions',

)

struct TweetPeriscopeContext {

1: bool is\_live = 0 (

strato.description='Indicates if the tweet contains live streaming video. A value of false is equivalent to this struct being undefined in the CreateTweetRequest.'

)

// Note that the REST API also defines a context\_periscope\_creator\_id param. The GraphQL

// API infers this value from the TwitterContext Viewer.userId since it should always be

// the same as the Tweet.coreData.userId which is also inferred from Viewer.userId.

} (

strato.description='Specifies information about live video streaming. Note that the Periscope product was shut down in March 2021, however some live video streaming features remain in the Twitter app. This struct keeps the Periscope naming convention to retain parity and traceability to other areas of the codebase that also retain the Periscope name.',

strato.graphql.typename='TweetPeriscopeContext',

)

struct TrustedFriendsControlOptions {

1: required i64 trusted\_friends\_list\_id (

strato.json.numbers.type='string',

strato.description='The ID of the Trusted Friends List whose members can view this tweet.'

)

} (

strato.description='Specifies information for a Trusted Friends tweet. See go/trusted-friends',

strato.graphql.typename='TrustedFriendsControlOptions',

)

enum CollabControlType {

COLLAB\_INVITATION = 1 (strato.description='This represents a CollabInvitation.')

// Note that a CollabTweet cannot be created through external graphql request,

// rather a user can create a CollabInvitation (which is automatically nullcasted) and a

// public CollabTweet will be created when all Collaborators have accepted the CollabInvitation,

// triggering a strato column to instantiate the CollabTweet directly

}(

strato.graphql.typename='CollabControlType',

)

struct CollabControlOptions {

1: required CollabControlType collabControlType

2: required list<i64> collaborator\_user\_ids (

strato.json.numbers.type='string',

strato.description='A list of user ids representing all Collaborators on a CollabTweet or CollabInvitation')

}(

strato.graphql.typename='CollabControlOptions',

strato.description='Specifies information about a CollabTweet or CollabInvitation (a union is used to ensure CollabControl defines one or the other). See more at go/collab-tweets.'

)

struct NoteTweetOptions {

1: required i64 note\_tweet\_id (

strato.json.numbers.type='string',

strato.description='The ID of the Note Tweet that has to be associated with the created Tweet.')

// Deprecated

2: optional list<string> mentioned\_screen\_names (

strato.description = 'Screen names of the users mentioned in the NoteTweet. This is used to set conversation control on the Tweet.')

3: optional list<i64> mentioned\_user\_ids (

strato.description = 'User ids of mentioned users in the NoteTweet. This is used to set conversation control on the Tweet, send mentioned user ids to TLS'

)

4: optional bool is\_expandable (

strato.description = 'Specifies if the Tweet can be expanded into the NoteTweet, or if they have the same text'

)

} (

strato.graphql.typename='NoteTweetOptions',

strato.description='Note Tweet options for a Tweet.'

)

// NOTE: Some clients were using the dark\_request directive in GraphQL to signal that a Tweet should not be persisted

// but this is not recommended, since the dark\_request directive is not meant to be used for business logic.

struct UndoOptions {

1: required bool is\_undo (

strato.description='Set to true if the Tweet is undo-able. Tweetypie will process the Tweet but will not persist it.'

)

} (

strato.graphql.typename='UndoOptions'

)

struct CreateTweetRequest {

1: string tweet\_text = "" (

strato.description='The user-supplied text of the tweet. Defaults to empty string. Leading & trailing whitespace are trimmed, remaining value may be empty if and only if one or more media entity ids are also provided.'

)

// @see com.twitter.tweetypie.thriftscala.PostTweetRequest.nullcast

2: bool nullcast = 0 (

strato.description='Do not deliver this tweet to a user\'s followers. http://go/nullcast'

)

// @see com.twitter.tweetypie.thriftscala.PostTweetRequest.PostTweetRequest.comparison\_id

3: optional string comparison\_id (

strato.description='Correlates requests originating from REST endpoints and GraphQL endpoints. UUID v4 (random) 36 character string.'

)

// @see com.twitter.ads.callback.thriftscala.EngagementRequest

4: optional engagement\_request.EngagementRequest engagement\_request (

strato.description='The ad engagement from which this tweet was created.'

)

// @see com.twitter.tweetypie.thriftscala.PostTweetRequest.attachment\_url

5: optional string attachment\_url (

strato.description='Tweet permalink (i.e. Quoted Tweet) or Direct Message deep link. This URL is not included in the visible\_text\_range.'

)

// @see com.twitter.tweetypie.thriftscala.Tweet.card\_reference

6: optional string card\_uri (

strato.description='Link to the card to associate with a tweet.'

)

7: optional TweetReply reply (

strato.description='Reply parameters.'

)

8: optional TweetMedia media (

strato.description='Media parameters.'

)

9: optional list<TweetAnnotation> semantic\_annotation\_ids (

strato.description='Escherbird Annotations.'

)

10: optional TweetGeo geo (

strato.description='Tweet geo location metadata. See https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/geo'

)

11: optional BatchComposeMode batch\_compose (

strato.description='Batch Compose Mode. See go/batchcompose'

)

12: optional ExclusiveTweetControlOptions exclusive\_tweet\_control\_options (

strato.description='When defined, this tweet will be marked as exclusive. Leave undefined to signify a regular, non-exclusive tweet. See go/superfollows.'

)

13: optional TweetConversationControl conversation\_control (

strato.description='Restrict replies to this tweet. See http://go/dont-at-me-api. Only valid for conversation root tweets. Applies to all replies to this tweet.'

)

14: optional TweetPeriscopeContext periscope (

strato.description='Specifies information about live video streaming. Note that the Periscope product was shut down in March 2021, however some live video streaming features remain in the Twitter app. This struct keeps the Periscope naming convention to retain parity and traceability to other areas of the codebase that also retain the Periscope name. Note: A value of periscope.isLive=false is equivalent to this struct being left undefined.'

)

15: optional TrustedFriendsControlOptions trusted\_friends\_control\_options (

strato.description='Trusted Friends parameters.'

)

16: optional CollabControlOptions collab\_control\_options (

strato.description='Collab Tweet & Collab Invitation parameters.'

)

17: optional EditOptions edit\_options (

strato.description='when defined, this tweet will be marked as an edit of the tweet represented by previous\_tweet\_id in edit\_options.'

)

18: optional NoteTweetOptions note\_tweet\_options (

strato.description='The Note Tweet that is to be associated with the created Tweet.',

strato.graphql.skip='true'

)

19: optional UndoOptions undo\_options (

strato.description='If the user has Undo Tweets enabled, the Tweet is created so that it can be previewed by the client but is not persisted.',

)

} (strato.graphql.typename='CreateTweetRequest')

struct CreateTweetResponse {

1: optional i64 tweet\_id (

strato.space='Tweet',

strato.graphql.fieldname='tweet',

strato.description='The created tweet.'

)

} (strato.graphql.typename='CreateTweetResponse')

struct CreateTweetResponseWithSubqueryPrefetchItems {

1: optional CreateTweetResponse data

2: optional existsAndPrefetch.PrefetchedData subqueryPrefetchItems

}

// Request struct, ResponseStruct, ResponseWithPrefetchStruct

struct DeleteTweetRequest {

1: required i64 tweet\_id (strato.json.numbers.type='string')

// @see com.twitter.tweetypie.thriftscala.PostTweetRequest.PostTweetRequest.comparison\_id

2: optional string comparison\_id (

strato.description='Correlates requests originating from REST endpoints and GraphQL endpoints. UUID v4 (random) 36 character string.'

)

} (strato.graphql.typename='DeleteTweetRequest')

struct DeleteTweetResponse {

1: optional i64 tweet\_id (

strato.space='Tweet',

strato.graphql.fieldname='tweet',

strato.description='The deleted Tweet. Since the Tweet will always be not found after deletion, the TweetResult will always be empty.'

)

} (strato.graphql.typename='DeleteTweetResponse')

struct DeleteTweetResponseWithSubqueryPrefetchItems {

1: optional DeleteTweetResponse data

2: optional existsAndPrefetch.PrefetchedData subqueryPrefetchItems

}