namespace java com.twitter.tweetypie.thriftjava

#@namespace scala com.twitter.tweetypie.thriftscala

#@namespace strato com.twitter.tweetypie

namespace py gen.twitter.tweetypie.tweet

namespace rb TweetyPie

// Specific namespace to avoid golang circular import

namespace go tweetypie.tweet

include "com/twitter/escherbird/tweet\_annotation.thrift"

include "com/twitter/expandodo/cards.thrift"

include "com/twitter/content-health/toxicreplyfilter/filtered\_reply\_details.thrift"

include "com/twitter/dataproducts/enrichments\_profilegeo.thrift"

include "com/twitter/geoduck/public/thriftv1/geoduck\_common.thrift"

include "com/twitter/mediaservices/commons/MediaCommon.thrift"

include "com/twitter/mediaservices/commons/MediaInformation.thrift"

include "com/twitter/tweetypie/api\_fields.thrift"

include "com/twitter/tweetypie/edit\_control.thrift"

include "com/twitter/tweetypie/media\_entity.thrift"

include "com/twitter/tweetypie/note\_tweet.thrift"

include "com/twitter/service/scarecrow/gen/tiered\_actions.thrift"

include "com/twitter/spam/rtf/safety\_label.thrift"

include "com/twitter/timelines/self\_thread/self\_thread.thrift"

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

include "com/twitter/tweet\_pivots/tweet\_pivots.thrift"

include "com/twitter/tweetypie/geo/tweet\_location\_info.thrift"

include "com/twitter/tweetypie/media/media\_ref.thrift"

include "unified\_cards\_contract.thrift"

include "com/twitter/tweetypie/creative-entity-enrichments/creative\_entity\_enrichments.thrift"

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

/\*\*

\* IDs are annotated with their corresponding space for Strato.

\*/

/\*\*

\* A Reply is data about a tweet in response to another tweet or a

\* user.

\*

\* This struct will be present if:

\* 1. This tweet is a reply to another tweet, or

\* 2. This tweet is directed at a user (the tweet's text begins with

\* an @mention).

\*/

struct Reply {

/\*\*

\* The id of the tweet that this tweet is replying to.

\*

\* This field will be missing for directed-at tweets (tweets whose

\* text begins with an @mention) that are not replying to another

\* tweet.

\*/

1: optional i64 in\_reply\_to\_status\_id (strato.space = "Tweet", strato.name = "inReplyToStatus", personalDataType = 'TweetId', tweetEditAllowed='false')

/\*\*

\* The user to whom this tweet is directed.

\*

\* If in\_reply\_to\_status\_id is set, this field is the author of that tweet.

\* If in\_reply\_to\_status\_id is not set, this field is the user mentioned at

\* the beginning of the tweet.

\*/

2: i64 in\_reply\_to\_user\_id (strato.space = "User", strato.name = "inReplyToUser", personalDataType = 'UserId')

/\*\*

\* The current username of in\_reply\_to\_user\_id.

\*

\* This field is not set when Gizmoduck returns a failure to Tweetypie.

\*/

3: optional string in\_reply\_to\_screen\_name (personalDataType = 'Username')

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

/\*\*

\* Includes information about the user a tweet is directed at (when a tweet

\* begins with @mention).

\*

\* Tweets with a DirectedAtUser are delivered to users who follow both the

\* author and the DirectedAtUser. Normally the DirectedAtUser will be the same

\* as Reply.in\_reply\_to\_user\_id, but will be different if the tweet's author

\* rearranges the @mentions in a reply.

\*/

struct DirectedAtUser {

1: i64 user\_id (strato.space = "User", strato.name = "user", personalDataType = 'UserId')

2: string screen\_name (personalDataType = 'Username')

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

/\*\*

\* A Share is data about the source tweet of a retweet.

\*

\* Share was the internal name for the retweet feature.

\*/

struct Share {

/\*\*

\* The id of the original tweet that was retweeted.

\*

\* This is always a tweet and never a retweet (unlike parent\_status\_id).

\*/

1: i64 source\_status\_id (strato.space = "Tweet", strato.name = "sourceStatus", personalDataType = 'TweetId')

/\*

\* The user id of the original tweet's author.

\*/

2: i64 source\_user\_id (strato.space = "User", strato.name = "sourceUser", personalDataType = 'UserId')

/\*\*

\* The id of the tweet that the user retweeted.

\*

\* Often this is the same as source\_status\_id, but it is different when a

\* user retweets via another retweet. For example, user A posts tweet id 1,

\* user B retweets it, creating tweet 2. If user user C sees B's retweet and

\* retweets it, the result is another retweet of tweet id 1, with the parent

\* status id of tweet 2.

\*/

3: i64 parent\_status\_id (strato.space = "Tweet", strato.name = "parentStatus", personalDataType = 'TweetId')

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

/\*\*

\* A record mapping a shortened URL (usually t.co) to a long url, and a prettified

\* display text. This is similar to data found in UrlEntity, and may replace that

\* data in the future.

\*/

struct ShortenedUrl {

/\*\*

\* Shortened t.co URL.

\*/

1: string short\_url (personalDataType = 'ShortUrl')

/\*\*

\* Original, full-length URL.

\*/

2: string long\_url (personalDataType = 'LongUrl')

/\*\*

\* Truncated version of expanded URL that does not include protocol and is

\* limited to 27 characters.

\*/

3: string display\_text (personalDataType = 'LongUrl')

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

/\*\*

\* A QuotedTweet is data about a tweet referenced within another tweet.

\*

\* QuotedTweet is included if Tweet.QuotedTweetField is requested, and the

\* linked-to tweet is public and visible at the time that the linking tweet

\* is hydrated, which can be during write-time or later after a cache-miss

\* read. Since linked-to tweets can be deleted, and users can become

\* suspended, deactivated, or protected, the presence of this value is not a

\* guarantee that the quoted tweet is still public and visible.

\*

\* Because a tweet quoting another tweet may not require a permalink URL in

\* the tweet's text, the URLs in ShortenedUrl may be useful to clients that

\* require maintaining a legacy-rendering of the tweet's text with the permalink.

\* See ShortenedUrl for details. Clients should avoid reading permalink whenever

\* possible and prefer the QuotedTweet's tweet\_id and user\_id instead.

\*

\* we always populate the permalink on tweet hydration unless there are partial

\* hydration errors or inner quoted tweet is filtered due to visibility rules.

\*

\*/

struct QuotedTweet {

1: i64 tweet\_id (strato.space = "Tweet", strato.name = "tweet", personalDataType = 'TweetId')

2: i64 user\_id (strato.space = "User", strato.name = "user", personalDataType = 'UserId')

3: optional ShortenedUrl permalink // URLs to access the quoted-tweet

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

/\*\*

\* A Contributor is a user who has access to another user's account.

\*/

struct Contributor {

1: i64 user\_id (strato.space = "User", strato.name = "user", personalDataType = 'UserId')

2: optional string screen\_name (personalDataType = 'Username')// not set on Gizmoduck failure

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

struct GeoCoordinates {

1: double latitude (personalDataType = 'GpsCoordinates')

2: double longitude (personalDataType = 'GpsCoordinates')

3: i32 geo\_precision = 0 (personalDataType = 'GpsCoordinates')

/\*\*

\* Whether or not make the coordinates public.

\*

\* This parameter is needed because coordinates are not typically published

\* by the author. If false: A tweet has geo coordinates shared but not make

\* it public.

\*/

4: bool display = 1

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

enum PlaceType {

UNKNOWN = 0

COUNTRY = 1

ADMIN = 2

CITY = 3

NEIGHBORHOOD = 4

POI = 5

}

enum PlaceNameType {

NORMAL = 0

ABBREVIATION = 1

SYNONYM = 2

}

struct PlaceName {

1: string name

2: string language = ""

3: PlaceNameType type

4: bool preferred

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

/\*\*

\* A Place is the physical and political properties of a location on Earth.

\*/

struct Place {

/\*\*

\* Geo service identifier.

\*/

1: string id (personalDataType = 'PublishedPreciseLocationTweet, PublishedCoarseLocationTweet')

/\*\*

\* Granularity of place.

\*/

2: PlaceType type

/\*\*

\* The name of this place composed with its parent locations.

\*

\* For example, the full name for "Brooklyn" would be "Brooklyn, NY". This

\* name is returned in the language specified by

\* GetTweetOptions.language\_tag.

\*/

3: string full\_name (personalDataType = 'InferredLocation')

/\*\*

\* The best name for this place as determined by geoduck heuristics.

\*

\* This name is returned in the language specified by

\* GetTweetOptions.language\_tag.

\*

\* @see com.twitter.geoduck.util.primitives.bestPlaceNameMatchingFilter

\*/

4: string name (personalDataType = 'PublishedPreciseLocationTweet, PublishedCoarseLocationTweet')

/\*\*

\* Arbitrary key/value data from the geoduck PlaceAttributes for this place.

\*/

5: map<string, string> attributes (personalDataTypeKey = 'PostalCode')

7: set<PlaceName> names

/\*\*

\* The ISO 3166-1 alpha-2 code for the country containing this place.

\*/

9: optional string country\_code (personalDataType = 'PublishedCoarseLocationTweet')

/\*\*

\* The best name for the country containing this place as determined by

\* geoduck heuristics.

\*

\* This name is returned in the language specified by

\* GetTweetOptions.language\_tag.

\*/

10: optional string country\_name (personalDataType = 'PublishedCoarseLocationTweet')

/\*\*

\* A simplified polygon that encompasses the place's geometry.

\*/

11: optional list<GeoCoordinates> bounding\_box

/\*\*

\* An unordered list of geo service identifiers for places that contain this

\* one from the most immediate parent up to the country.

\*/

12: optional set<string> containers (personalDataType = 'PublishedCoarseLocationTweet')

/\*\*

\* A centroid-like coordinate that is within the geometry of the place.

\*/

13: optional GeoCoordinates centroid

/\*\*

\* Reason this place is being suppressed from display.

\*

\* This field is present when we previously had a place for this ID, but are

\* now choosing not to hydrate it and instead providing fake place metadata

\* along with a reason for not including place information.

\*/

14: optional geoduck\_common.WithheldReason withheldReason

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

/\*\*

\* A UrlEntity is the position and content of a t.co shortened URL in the

\* tweet's text.

\*

\* If Talon returns an error to Tweetypie during tweet hydration, the

\* UrlEntity will be omitted from the response. UrlEntities are not included

\* for non-t.co-wrapped URLs found in older tweets, for spam and user safety

\* reasons.

\*/

struct UrlEntity {

/\*\*

\* The position of this entity's first character, in zero-indexed Unicode

\* code points.

\*/

1: i16 from\_index

/\*\*

\* The position after this entity's last character, in zero-indexed Unicode

\* code points.

\*/

2: i16 to\_index

/\*\*

\* Shortened t.co URL.

\*/

3: string url (personalDataType = 'ShortUrl')

/\*\*

\* Original, full-length URL.

\*

\* This field will always be present on URL entities returned by

\* Tweetypie; it is optional as an implementation artifact.

\*/

4: optional string expanded (personalDataType = 'LongUrl')

/\*\*

\* Truncated version of expanded URL that does not include protocol and is

\* limited to 27 characters.

\*

\* This field will always be present on URL entities returned by

\* Tweetypie; it is optional as an implementation artifact.

\*/

5: optional string display (personalDataType = 'LongUrl')

6: optional i64 click\_count (personalDataType = 'CountOfTweetEntitiesClicked')

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

/\*\*

\* A MentionEntity is the position and content of a mention, (the "@"

\* character followed by the name of another valid user) in a tweet's text.

\*

\* If Gizmoduck returns an error to Tweetypie during tweet hydration that

\* MentionEntity will be omitted from the response.

\*/

struct MentionEntity {

/\*\*

\* The position of this entity's first character ("@"), in zero-indexed

\* Unicode code points.

\*/

1: i16 from\_index

/\*\*

\* The position after this entity's last character, in zero-indexed Unicode

\* code points.

\*/

2: i16 to\_index

/\*\*

\* Contents of the mention without the leading "@".

\*/

3: string screen\_name (personalDataType = 'Username')

/\*\*

\* User id of the current user with the mentioned screen name.

\*

\* In the current implementation user id does not necessarily identify the

\* user who was originally mentioned when the tweet was created, only the

\* user who owns the mentioned screen name at the time of hydration. If a

\* mentioned user changes their screen name and a second user takes the old

\* name, this field identifies the second user.

\*

\* This field will always be present on mention entities returned by

\* Tweetypie; it is optional as an implementation artifact.

\*/

4: optional i64 user\_id (strato.space = "User", strato.name = "user", personalDataType = 'UserId')

/\*\*

\* Display name of the current user with the mentioned screen name.

\*

\* See user\_id for caveats about which user's name is used here. This field

\* will always be present on mention entities returned by Tweetypie; it is

\* optional as an implementation artifact.

\*/

5: optional string name (personalDataType = 'DisplayName')

/\*\*

\* Indicates if the user referred to by this MentionEntity has been unmentioned

\* from the conversation. If this field is set to true, the fromIndex and toIndex

\* fields will have a value of 0.

\*

\* @deprecated isUnmentioned is no longer being populated

\*/

6: optional bool isUnmentioned (personalDataType = 'ContentPrivacySettings')

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

/\*\*

\* A list of users that are mentioned in the tweet and have a blocking

\* relationship with the tweet author. Mentions for these users will be unlinked

\* in the tweet.

\*/

struct BlockingUnmentions {

1: optional list<i64> unmentioned\_user\_ids (strato.space = 'User', strato.name = 'users', personalDataType = 'UserId')

}(persisted = 'true', hasPersonalData = 'true', strato.graphql.typename = 'BlockingUnmentions')

/\*\*

\* A list of users that are mentioned in the tweet and have indicated they do not want

\* to be mentioned via their mention settings. Mentions for these users will be unlinked

\* in the tweet by Twitter owned and operated clients.

\*/

struct SettingsUnmentions {

1: optional list<i64> unmentioned\_user\_ids (strato.space = 'User', strato.name = 'users', personalDataType = 'UserId')

}(persisted = 'true', hasPersonalData = 'true', strato.graphql.typename = 'SettingsUnmentions')

/\*\*

\* A HashtagEntity is the position and content of a hashtag (a term starting

\* with "#") in a tweet's text.

\*/

struct HashtagEntity {

/\*\*

\* The position of this entity's first character ("#"), in zero-indexed

\* Unicode code points.

\*/

1: i16 from\_index

/\*\*

\* The position after this entity's last character, in zero-indexed Unicode

\* code points.

\*/

2: i16 to\_index

/\*\*

\* Contents of the hashtag without the leading "#".

\*/

3: string text (personalDataType = 'PrivateTweetEntitiesAndMetadata, PublicTweetEntitiesAndMetadata')

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

/\*\*

\* A CashtagEntity is the position and content of a cashtag (a term starting

\* with "$") in a tweet's text.

\*/

struct CashtagEntity {

/\*\*

\* The position of this entity's first character, in zero-indexed Unicode

\* code points.

\*/

1: i16 from\_index

/\*\*

\* The position after this entity's last character, in zero-indexed Unicode

\* code points.

\*/

2: i16 to\_index

/\*\*

\* Contents of the cashtag without the leading "$"

\*/

3: string text (personalDataType = 'PrivateTweetEntitiesAndMetadata, PublicTweetEntitiesAndMetadata')

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

enum MediaTagType {

USER = 0

RESERVED\_1 = 1

RESERVED\_2 = 2

RESERVED\_3 = 3

RESERVED\_4 = 4

}

struct MediaTag {

1: MediaTagType tag\_type

2: optional i64 user\_id (strato.space = "User", strato.name = "user", personalDataType = 'UserId')

3: optional string screen\_name (personalDataType = 'Username')

4: optional string name (personalDataType = 'DisplayName')

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

struct TweetMediaTags {

1: map<MediaCommon.MediaId, list<MediaTag>> tag\_map

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

/\*\*

\* A UserMention is a user reference not stored in the tweet text.

\*

\* @deprecated Was used only in ReplyAddresses

\*/

struct UserMention {

1: i64 user\_id (strato.space = "User", strato.name = "user", personalDataType = 'UserId')

2: optional string screen\_name (personalDataType = 'Username')

3: optional string name (personalDataType = 'DisplayName')

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

/\*\*

\* ReplyAddresses is a list of reply entities which are stored outside of the

\* text.

\*

\* @deprecated

\*/

struct ReplyAddresses {

1: list<UserMention> users = []

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

/\*\*

\* SchedulingInfo is metadata about tweets created by the tweet scheduling

\* service.

\*/

//

struct SchedulingInfo {

/\*\*

\* Id of the corresponding scheduled tweet before it was created as a real

\* tweet.

\*/

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

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

/\*\*

\* @deprecated

\*/

enum SuggestType {

WTF\_CARD = 0

WORLD\_CUP = 1

WTD\_CARD = 2

NEWS\_CARD = 3

RESERVED\_4 = 4

RESERVED\_5 = 5

RESERVED\_6 = 6

RESERVED\_7 = 7

RESERVED\_8 = 8

RESERVED\_9 = 9

RESERVED\_10 = 10

RESERVED\_11 = 11

}

/\*\*

\* @deprecated

\*/

enum TwitterSuggestsVisibilityType {

/\*\*

\* Always public to everyone

\*/

PUBLIC = 1

/\*\*

\* Inherits visibility rules of personalized\_for\_user\_id.

\*/

RESTRICTED = 2

/\*\*

\* Only visible to personalized\_for\_user\_id (and author).

\*/

PRIVATE = 3

}

/\*\*

\* TwitterSuggestInfo is details about a synthetic tweet generated by an early

\* version of Twitter Suggests.

\*

\* @deprecated

\*/

struct TwitterSuggestInfo {

1: SuggestType suggest\_type

2: TwitterSuggestsVisibilityType visibility\_type

3: optional i64 personalized\_for\_user\_id (strato.space = "User", strato.name = "personalizedForUser", personalDataType = 'UserId')

4: optional i64 display\_timestamp\_secs (personalDataType = 'PublicTimestamp')

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

/\*\*

\* A DeviceSource contains information about the client application from which

\* a tweet was sent.

\*

\* This information is stored in Passbird. The developer that owns a client

\* application provides this information on https://apps.twitter.com.

\*/

struct DeviceSource {

/\*\*

\* The id of the client in the now deprecated device\_sources MySQL table.

\*

\* Today this value will always be 0.

\*

\* @deprecated Use client\_app\_id

\*/

1: required i64 id (personalDataType = 'AppId')

/\*\*

\* Identifier for the client in the format "oauth:<client\_app\_id>"

\*/

2: string parameter

/\*\*

\* Identifier for the client in the format "oauth:<client\_app\_id>"

\*/

3: string internal\_name

/\*\*

\* Developer-provided name of the client application.

\*/

4: string name

/\*\*

\* Developer-provided publicly accessible home page for the client

\* application.

\*/

5: string url

/\*\*

\* HTML fragment with a link to the client-provided URL

\*/

6: string display

/\*\*

\* This field is marked optional for backwards compatibility but will always

\* be populated by Tweetypie.

\*/

7: optional i64 client\_app\_id (personalDataType = 'AppId')

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

/\*\*

\* A Narrowcast restricts delivery of a tweet geographically.

\*

\* Narrowcasts allow multi-national advertisers to create geo-relevant content

\* from a central handle that is only delivered to to followers in a

\* particular country or set of countries.

\*/

struct Narrowcast {

2: list<string> location = [] (personalDataType = 'PublishedCoarseLocationTweet')

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

/\*\*

\* StatusCounts is a summary of engagement metrics for a tweet.

\*

\* These metrics are loaded from TFlock.

\*/

struct StatusCounts {

/\*\*

\* Number of times this tweet has been retweeted.

\*

\* This number may not match the list of users who have retweeted because it

\* includes retweets from protected and suspended users who are not listed.

\*/

1: optional i64 retweet\_count (personalDataType = 'CountOfPrivateRetweets, CountOfPublicRetweets', strato.json.numbers.type = 'int53')

/\*\*

\* Number of direct replies to this tweet.

\*

\* This number does not include replies to replies.

\*/

2: optional i64 reply\_count (personalDataType = 'CountOfPrivateReplies, CountOfPublicReplies', strato.json.numbers.type = 'int53')

/\*\*

\* Number of favorites this tweet has received.

\*

\* This number may not match the list of users who have favorited a tweet

\* because it includes favorites from protected and suspended users who are

\* not listed.

\*/

3: optional i64 favorite\_count (personalDataType = 'CountOfPrivateLikes, CountOfPublicLikes', strato.json.numbers.type = 'int53')

/\*\*

\* @deprecated

\*/

4: optional i64 unique\_users\_impressed\_count (strato.json.numbers.type = 'int53')

/\*\*

\* Number of replies to this tweet including replies to replies.

\*

\* @deprecated

\*/

5: optional i64 descendent\_reply\_count (personalDataType = 'CountOfPrivateReplies, CountOfPublicReplies', strato.json.numbers.type = 'int53')

/\*\*

\* Number of times this tweet has been quote tweeted.

\*

\* This number may not match the list of users who have quote tweeted because it

\* includes quote tweets from protected and suspended users who are not listed.

\*/

6: optional i64 quote\_count (personalDataType = 'CountOfPrivateRetweets, CountOfPublicRetweets', strato.json.numbers.type = 'int53')

/\*\*

\* Number of bookmarks this tweet has received.

\*/

7: optional i64 bookmark\_count (personalDataType = 'CountOfPrivateLikes', strato.json.numbers.type = 'int53')

}(persisted='true', hasPersonalData = 'true', strato.graphql.typename='StatusCounts')

/\*\*

\* A is a tweet's properties from one user's point of view.

\*/

struct StatusPerspective {

1: i64 user\_id (strato.space = "User", strato.name = "user", personalDataType = 'UserId')

/\*\*

\* Whether user\_id has favorited this tweet.

\*/

2: bool favorited

/\*\*

\* Whether user\_id has retweeted this tweet.

\*/

3: bool retweeted

/\*\*

\* If user\_id has retweeted this tweet, retweet\_id identifies that tweet.

\*/

4: optional i64 retweet\_id (strato.space = "Tweet", strato.name = "retweet", personalDataType = 'TweetId')

/\*\*

\* Whether user\_id has reported this tweet as spam, offensive, or otherwise

\* objectionable.

\*/

5: bool reported

/\*\*

\* Whether user\_id has bookmarked this tweet.

\*/

6: optional bool bookmarked

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

/\*\*

\* A Language is a guess about the human language of a tweet's text.

\*

\* Language is determined by TwitterLanguageIdentifier from the

\* com.twitter.common.text package (commonly called "Penguin").

\*/

struct Language {

/\*\*

\* Language code in BCP-47 format.

\*/

1: required string language (personalDataType = 'InferredLanguage')

/\*\*

\* Language direction.

\*/

2: bool right\_to\_left

/\*\*

\* Confidence level of the detected language.

\*/

3: double confidence = 1.0

/\*\*

\* Other possible languages and their confidence levels.

\*/

4: optional map<string, double> other\_candidates

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

/\*\*

\* A SupplementalLanguage is a guess about the human language of a tweet's

\* text.

\*

\* SupplementalLanguage is typically determined by a third-party translation

\* service. It is only stored when the service detects a different language

\* than TwitterLanguageIdentifier.

\*

\* @deprecated 2020-07-08 no longer populated.

\*/

struct SupplementalLanguage {

/\*\*

\* Language code in BCP-47 format.

\*/

1: required string language (personalDataType = 'InferredLanguage')

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

/\*\*

\* A SpamLabel is a collection of spam actions for a tweet.

\*

\* Absence of a SpamLabel indicates that no action needs to be taken

\*/

struct SpamLabel {

/\*\*

\* Filter this content at render-time

\*

\* @deprecated 2014-05-19 Use filter\_renders

\*/

1: bool spam = 0

2: optional set<tiered\_actions.TieredActionResult> actions;

}(persisted='true')

/\*\*

\* The available types of spam signal

\*

\* @deprecated

\*/

enum SpamSignalType {

MENTION = 1

SEARCH = 2

STREAMING = 4

# OBSOLETE HOME\_TIMELINE = 3

# OBSOLETE NOTIFICATION = 5

# OBSOLETE CONVERSATION = 6

# OBSOLETE CREATION = 7

RESERVED\_VALUE\_8 = 8

RESERVED\_VALUE\_9 = 9

RESERVED\_VALUE\_10 = 10

}

/\*\*

\* @deprecated

\* CardBindingValues is a collection of key-value pairs used to render a card.

\*/

struct CardBindingValues {

1: list<cards.Card2ImmediateValue> pairs = []

}(persisted='true')

/\*\*

\* A CardReference is a mechanism for explicitly associating a card with a

\* tweet.

\*/

struct CardReference {

/\*\*

\* Link to the card to associate with a tweet.

\*

\* This URI may reference either a card stored in the card service, or

\* another resource, such as a crawled web page URL. This value supercedes

\* any URL present in tweet text.

\*/

1: string card\_uri

}(persisted='true')

/\*\*

\* A TweetPivot is a semantic entity related to a tweet.

\*

\* TweetPivots are used to direct to the user to another related location. For

\* example, a "See more about <name>" UI element that takes the user to <url>

\* when clicked.

\*/

struct TweetPivot {

1: required tweet\_annotation.TweetEntityAnnotation annotation

2: required tweet\_pivots.TweetPivotData data

}(persisted='true')

struct TweetPivots {

1: required list<TweetPivot> tweet\_pivots

}(persisted='true')

struct EscherbirdEntityAnnotations {

1: list<tweet\_annotation.TweetEntityAnnotation> entity\_annotations

}(persisted='true')

struct TextRange {

/\*\*

\* The inclusive index of the start of the range, in zero-indexed Unicode

\* code points.

\*/

1: required i32 from\_index

/\*\*

\* The exclusive index of the end of the range, in zero-indexed Unicode

\* code points.

\*/

2: required i32 to\_index

}(persisted='true')

struct TweetCoreData {

1: i64 user\_id (strato.space = "User", strato.name = "user", personalDataType = 'UserId', tweetEditAllowed='false')

/\*\*

\* The body of the tweet consisting of the user-supplied displayable message

\* and:

\* - an optional prefix list of @mentions

\* - an optional suffix attachment url.

\*

\* The indices from visible\_text\_range specify the substring of text indended

\* to be displayed, whose length is limited to 140 display characters. Note

\* that the visible substring may be longer than 140 characters due to HTML

\* entity encoding of &, <, and > .

\* For retweets the text is that of the original tweet, prepended with "RT

\* @username: " and truncated to 140 characters.

\*/

2: string text (personalDataType = 'PrivateTweets, PublicTweets')

/\*\*

\* The client from which this tweet was created

\*

\* The format of this value is oauth:<client id>.

\*/

3: string created\_via (personalDataType = 'ClientType')

/\*\*

\* Time this tweet was created.

\*

\* This value is seconds since the Unix epoch. For tweets with Snowflake IDs

\* this value is redundant, since a millisecond-precision timestamp is part

\* of the id.

\*/

4: i64 created\_at\_secs

/\*\*

\* Present when this tweet is a reply to another tweet or another user.

\*/

5: optional Reply reply

/\*\*

\* Present when a tweet begins with an @mention or has metadata indicating the directed-at user.

\*/

6: optional DirectedAtUser directed\_at\_user

/\*\*

\* Present when this tweet is a retweet.

\*/

7: optional Share share

/\*\*

\* Whether there is a takedown country code or takedown reason set for this specific tweet.

\*

\* See takedown\_country\_codes for the countries where the takedown is active. (deprecated)

\* See takedown\_reasons for a list of reasons why the tweet is taken down.

\*

\* has\_takedown will be set to true if either this specific tweet or the author has a

\* takedown active.

\*/

8: bool has\_takedown = 0

/\*\*

\* Whether this tweet might be not-safe-for-work, judged by the tweet author.

\*

\* Users can flag their own accounts as not-safe-for-work in account

\* preferences by selecting "Mark media I tweet as containing material that

\* may be sensitive" and each tweet created after that point will have

\* this flag set.

\*

\* The value can also be updated after tweet create time via the

\* update\_possibly\_sensitive\_tweet method.

\*/

9: bool nsfw\_user = 0

/\*\*

\* Whether this tweet might be not-safe-for-work, judged by an internal Twitter

\* support agent.

\*

\* This tweet value originates from the user's nsfw\_admin flag at

\* tweet create time but can be updated afterwards using the

\* update\_possibly\_sensitive\_tweet method.

\*/

10: bool nsfw\_admin = 0

/\*\*

\* When nullcast is true a tweet is not delivered to a user's followers, not

\* shown in the user's timeline, and does not appear in search results.

\*

\* This is primarily used to create tweets that can be used as ads without

\* broadcasting them to an advertiser's followers.

\*/

11: bool nullcast = 0 (tweetEditAllowed='false')

/\*\*

\* Narrowcast limits delivery of a tweet to followers in specific geographic

\* regions.

\*/

12: optional Narrowcast narrowcast (tweetEditAllowed='false')

/\*\*

\* The impression id of the ad from which this tweet was created.

\*

\* This is set when a user retweets or replies to a promoted tweet. It is

\* used to attribute the "earned" exposure of an advertisement.

\*/

13: optional i64 tracking\_id (personalDataType = 'ImpressionId', tweetEditAllowed='false')

/\*\*

\* A shared identifier among all the tweets in the reply chain for a single

\* tweet.

\*

\* The conversation id is the id of the tweet that started the conversation.

\*/

14: optional i64 conversation\_id (strato.space = "Tweet", strato.name = "conversation", personalDataType = 'TweetId')

/\*\*

\* Whether this tweet has media of any type.

\*

\* Media can be in the form of media entities, media cards, or URLs in the

\* tweet text that link to media partners.

\*

\* @see MediaIndexHelper

\*/

15: optional bool has\_media

/\*\*

\* Supported for legacy clients to associate a location with a Tweet.

\*

\* Twitter owned clients must use place\_id REST API param for geo-tagging.

\*

\* @deprecated Use place\_id REST API param

\*/

16: optional GeoCoordinates coordinates (personalDataType = 'GpsCoordinates', tweetEditAllowed='false')

/\*\*

\* The location where a tweet was sent from.

\*

\* Place is either published in API request explicitly or implicitly reverse

\* geocoded from API lat/lon coordinates params.

\*

\* Tweetypie implementation notes:

\* - Currently, if both place\_id and coordinates are specified, coordinates

\* takes precedence in geo-tagging. I.e.: Place returned rgc(coordinates)

\* sets the place\_id field.

\* - place\_id is reverse geocoded on write-path.

\*/

17: optional string place\_id (personalDataType = 'PublishedPreciseLocationTweet, PublishedCoarseLocationTweet')

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

/\*\*

\* List of community ID's the tweet belongs to.

\*/

struct Communities {

1: required list<i64> community\_ids (personalDataType = 'EngagementId')

}(persisted='true')

/\*\*

\* Tweet metadata that is present on extended tweets, a tweet whose total text length is greater

\* than the classic limit of 140 characters.

\*/

struct ExtendedTweetMetadata {

/\*\*

\* @deprecated was display\_count

\*/

1: i32 unused1 = 0

/\*\*

\* The index, in unicode code points, at which the tweet text should be truncated

\* for rendering in a public API backwards-compatible mode. Once truncated to this

\* point, the text should be appended with an ellipsis, a space, and the short\_url

\* from self\_permalink. The resulting text must conform to the 140 display glyph

\* limit.

\*/

2: required i32 api\_compatible\_truncation\_index

/\*\*

\* @deprecated was default\_display\_truncation\_index

\*/

3: i32 unused3 = 0

/\*\*

\* @deprecated was is\_long\_form

\*/

4: bool unused4 = 0

/\*\*

\* @deprecated was preview\_range

\*/

5: optional TextRange unused5

/\*\*

\* @deprecated was extended\_preview\_range

\*/

6: optional TextRange unused6

}(persisted='true')

/\*\*

\* @deprecated use TransientCreateContext instead

\*/

enum TweetCreateContextKey {

PERISCOPE\_IS\_LIVE = 0,

PERISCOPE\_CREATOR\_ID = 1

}

/\*\*

\* DirectedAtUserMetadata is a tweetypie-internal structure that can be used to store metadata about

\* a directed-at user on the tweet.

\*

\* Note: absence of this field does not imply the tweet does not have a DirectedAtUser, see

\* tweet.directedAtUserMetadata for more information.

\*/

struct DirectedAtUserMetadata {

/\*\*

\* ID of the user a tweet is directed-at.

\*/

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

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

/\*\*

\* Tweet metadata that may be present on tweets in a self-thread (tweetstorm).

\*

\* A self-thread is a tree of self-replies that may either:

\* 1. begin as a reply to another user's tweet (called a non-root self-thread) or

\* 2. stand alone (called root self-thread).

\*

\* Note that not all self-threads have SelfThreadMetadata.

\*/

struct SelfThreadMetadata {

/\*\*

\* A shared identifier among all the tweets in the self-thread (tweetstorm).

\*

\* The tweetstorm id is the id of the tweet that started the self thread.

\*

\* If the id matches the tweet's conversation\_id then it is a root self-thread, otherwise it is

\* a non-root self-thread.

\*/

1: required i64 id (personalDataType = 'TweetId')

/\*\*

\* Indicates if the tweet with this SelfThreadMetadata is a leaf in the self-thread tree.

\* This flag might be used to encourage the author to extend their tweetstorm at the end.

\*/

2: bool isLeaf = 0

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

/\*\*

\* Composer flow used to create this tweet. Unless using the News Camera (go/newscamera)

\* flow, this should be `STANDARD`.

\*

\* When set to `CAMERA`, clients are expected to display the tweet with a different UI

\* to emphasize attached media.

\*/

enum ComposerSource {

STANDARD = 1

CAMERA = 2

}

/\*\*

\* The conversation owner and users in invited\_user\_ids can reply

\*\*/

struct ConversationControlByInvitation {

1: required list<i64> invited\_user\_ids (personalDataType = 'UserId')

2: required i64 conversation\_tweet\_author\_id (personalDataType = 'UserId')

3: optional bool invite\_via\_mention

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

/\*\*

\* The conversation owner, users in invited\_user\_ids, and users who the conversation owner follows can reply

\*\*/

struct ConversationControlCommunity {

1: required list<i64> invited\_user\_ids (personalDataType = 'UserId')

2: required i64 conversation\_tweet\_author\_id (personalDataType = 'UserId')

3: optional bool invite\_via\_mention

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

/\*\*

\* The conversation owner, users in invited\_user\_ids, and users who follows the conversation owner can reply

\*\*/

struct ConversationControlFollowers {

1: required list<i64> invited\_user\_ids (personalDataType = 'UserId')

2: required i64 conversation\_tweet\_author\_id (personalDataType = 'UserId')

3: optional bool invite\_via\_mention

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

/\*\*

\* This tweet metadata captures restrictions on who is allowed to reply in a conversation.

\*/

union ConversationControl {

1: ConversationControlCommunity community

2: ConversationControlByInvitation byInvitation

3: ConversationControlFollowers followers

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

// This tweet metadata shows the exclusivity of a tweet and is used to determine

// whether replies / visibility of a tweet is limited

struct ExclusiveTweetControl {

1: required i64 conversation\_author\_id (personalDataType = 'UserId')

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

/\*\*

\* Tweet metadata for a Trusted Friends tweet.

\*

\* A Trusted Friends tweet is a tweet whose visibility is restricted to members

\* of an author-specified list.

\*

\* Replies to a Trusted Friends tweet will inherit a copy of this metadata from

\* the root tweet.

\*/

struct TrustedFriendsControl {

/\*\*

\* The ID of the Trusted Friends List whose members can view this tweet.

\*/

1: required i64 trusted\_friends\_list\_id (personalDataType = 'TrustedFriendsListMetadata')

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

enum CollabInvitationStatus {

PENDING = 0

ACCEPTED = 1

REJECTED = 2

}

/\*\*

\* Represents a user who has been invited to collaborate on a CollabTweet, associated with whether

\* they have accepted or rejected collaboration

\*/

struct InvitedCollaborator {

1: required i64 collaborator\_user\_id (personalDataType = 'UserId')

2: required CollabInvitationStatus collab\_invitation\_status

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

/\*\*

\* Present if Tweet is a CollabInvitation awaiting publishing, stores list of invited Collaborators

\*/

struct CollabInvitation {

1: required list<InvitedCollaborator> invited\_collaborators

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

/\*\*

\* Present if Tweet is a published CollabTweet, stores list of Collaborators

\*/

struct CollabTweet {

1: required list<i64> collaborator\_user\_ids (personalDataType = 'UserId')

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

/\*\*

\* CollabTweets treat multiple users as co-authors or "Collaborators" of a single "Collab Tweet".

\*

\* When creating a Collab Tweet, the original author will begin by creating a CollabInvitation which

\* is sent to another Collaborator to accept or reject collaboration. If and when other

\* Collaborators have accepted, the CollabInvitation is replaced by a CollabTweet which is published

\* publicly and fanned out to followers of all Collaborators. A CollabInvitation will be hidden from

\* anyone except the list of Collaborators using VF. The CollabTweet will then be fanned out like

\* a regular Tweet to the profiles and combined audiences of all Collaborators.

\*

\* A Tweet representing a CollabTweet or CollabInvitation is denoted by the presence of a

\* CollabControl field on a Tweet.

\*/

union CollabControl {

1: CollabInvitation collab\_invitation

2: CollabTweet collab\_tweet

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

/\*\*

\* A Tweet is a message that belongs to a Twitter user.

\*

\* The Tweet struct replaces the deprecated Status struct. All fields except

\* id are optional.

\*

\* This struct supports the additional fields flexible schema. Additional fields are

\* defined starting from field 101.

\*

\* The guidelines for adding a new Additional field:

\* 1. It's required to define the additional field as an optional struct.

\* Inside the struct, define optional or non-optional field(s) according

\* to your needs.

\* 2. If you have several immutable piece of data that are always accessed

\* together, you should define them in the same struct for better storage

\* locality.

\* 3. If your data model has several mutable pieces, and different piece can

\* be updated in a close succession, you should group them into

\* separate structs and each struct contains one mutable piece.

\*/

struct Tweet {

/\*\*

\* The primary key for a tweet.

\*

\* A tweet's id is assigned by the tweet service at creation time. Since

\* 2010-11-04 tweet ids have been generated using Snowflake. Prior to this

\* ids were assigned sequentially by MySQL AUTOINCREMENT.

\*/

1: i64 id (personalDataType = 'TweetId')

/\*\*

\* The essential properties of a tweet.

\*

\* This field will always be present on tweets returned by Tweetypie. It is

\* marked optional so an empty tweet can be provided to write additional

\* fields.

\*/

2: optional TweetCoreData core\_data

/\*\*

\* URLs extracted from the tweet's text.

\*/

3: optional list<UrlEntity> urls

/\*\*

\* Mentions extracted from the tweet's text.

\*/

4: optional list<MentionEntity> mentions

/\*\*

\* Hashtags extracted from the tweet's text.

\*/

5: optional list<HashtagEntity> hashtags

/\*\*

\* Cashtags extracted from the tweet's text

\*/

6: optional list<CashtagEntity> cashtags

7: optional list<media\_entity.MediaEntity> media

/\*\*

\* Place identified by Tweet.core\_data.place\_id.

\*/

10: optional Place place

11: optional QuotedTweet quoted\_tweet

/\*\*

\* The list of countries where this tweet will not be shown.

\*

\* This field contains countries for both the tweet and the user, so it may

\* contain values even if has\_takedown is false.

\*

\* @deprecated, use field 30 takedown\_reasons which includes the same information and more

\*/

12: optional list<string> takedown\_country\_codes (personalDataType = 'ContentRestrictionStatus')

/\*\*

\* Interaction metrics for this tweet.

\*

\* Included when one of GetTweetOptions.load\_retweet\_count,

\* GetTweetOptions.load\_reply\_count, or GetTweetOptions.load\_favorite\_count

\* is set. This can be missing in a PARTIAL response if the TFlock request

\* fails.

\*/

13: optional StatusCounts counts

/\*\*

\* Properties of the client from which the tweet was sent.

\*

\* This can be missing in a PARTIAL response if the Passbird request fails.

\*/

14: optional DeviceSource device\_source

/\*\*

\* Properties of this tweet from the point of view of

\* GetTweetOptions.for\_user\_id.

\*

\* This field is included only when for\_user\_id is provided and

\* include\_perspective == true This can be missing in a PARTIAL response if

\* the timeline service request fails.

\*/

15: optional StatusPerspective perspective

/\*\*

\* Version 1 cards.

\*

\* This field is included only when GetTweetOptions.include\_cards == true.

\*/

16: optional list<cards.Card> cards

/\*\*

\* Version 2 cards.

\*

\* This field is included only included when GetTweetOptions.include\_cards

\* == true and GetTweetOptions.cards\_platform\_key is set to valid value.

\*/

17: optional cards.Card2 card2

/\*\*

\* Human language of tweet text as determined by TwitterLanguageIdentifier.

\*/

18: optional Language language

/\*\*

\* @deprecated

\*/

19: optional map<SpamSignalType, SpamLabel> spam\_labels

/\*\*

\* User responsible for creating this tweet when it is not the same as the

\* core\_data.user\_id.

\*

\* This is sensitive information and must not be shared externally (via UI,

\* API, or streaming) except to the the owner of the tweet

\* (core\_data.user\_id) or a contributor to the owner's account.

\*/

20: optional Contributor contributor

// obsolete 21: optional list<TopicLabel> topic\_labels

22: optional enrichments\_profilegeo.ProfileGeoEnrichment profile\_geo\_enrichment

// Maps extension name to value; only populated if the request contained an extension on tweets.

// obsolete 24: optional map<string, binary> extensions

/\*\*

\* Deprecated.

\* Semantic entities that are related to this tweet.

\*/

25: optional TweetPivots tweet\_pivots

/\*\*

\* @deprecated

\* Strato Tweet Extensions support has moved to birdherd.

\*

\* Internal thrift clients should query strato columns directly and

\* not rely upon ext/\*.Tweet columns which are designed to serve

\* client APIs.

\*/

26: optional binary extensions\_reply

/\*\*

\* Has the requesting user muted the conversation referred to by

\* `conversation\_id`? When this field is absent, the conversation may

\* or may not be muted. Use the `include\_conversation\_muted` field in

\* GetTweetOptions to request this field.

\*

\* If this field has a value, the value applies to the user in the

\* `for\_user\_id` field of the requesting `GetTweetOptions`.

\*/

27: optional bool conversation\_muted

/\*\*

\* The user id of the tweet referenced by conversation\_id

\*

\* @deprecated Was conversation\_owner\_id. This was never implemented.

\*/

28: optional i64 unused28

/\*\*

\* Has this tweet been removed from its conversation by the conversation owner?

\*

\* @deprecated Was is\_removed\_from\_conversation. This was never implemented.

\*/

29: optional bool unused29

/\*\*

\* A list of takedown reasons indicating which country and reason this tweet was taken down.

\*/

30: optional list<withholding.TakedownReason> takedown\_reasons

/\*\*

\* @obsolete, self-thread metadata is now stored in field 151, self\_thread\_metadata

\*/

31: optional self\_thread.SelfThreadInfo self\_thread\_info

// field 32 to 99 are reserved

// field 100 is used for flexible schema proof of concept

// additional fields

// these fields are stored in Manhattan flexible schema

101: optional TweetMediaTags media\_tags

102: optional SchedulingInfo scheduling\_info

/\*\*

\* @deprecated

\*/

103: optional CardBindingValues binding\_values

/\*\*

\* @deprecated

\*/

104: optional ReplyAddresses reply\_addresses

/\*\*

\* OBSOLETE, but originally contained information about synthetic tweets created by the first

\* version of Twitter Suggests.

\*

\* @deprecated

\*/

105: optional TwitterSuggestInfo obsolete\_twitter\_suggest\_info

106: optional EscherbirdEntityAnnotations escherbird\_entity\_annotations (personalDataType = 'AnnotationValue')

// @deprecated 2021-07-19

107: optional safety\_label.SafetyLabel spam\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

108: optional safety\_label.SafetyLabel abusive\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

109: optional safety\_label.SafetyLabel low\_quality\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

110: optional safety\_label.SafetyLabel nsfw\_high\_precision\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

111: optional safety\_label.SafetyLabel nsfw\_high\_recall\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

112: optional safety\_label.SafetyLabel abusive\_high\_recall\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

113: optional safety\_label.SafetyLabel low\_quality\_high\_recall\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

114: optional safety\_label.SafetyLabel persona\_non\_grata\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

115: optional safety\_label.SafetyLabel recommendations\_low\_quality\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

116: optional safety\_label.SafetyLabel experimentation\_label (personalDataType = 'TweetSafetyLabels')

117: optional tweet\_location\_info.TweetLocationInfo tweet\_location\_info

118: optional CardReference card\_reference

/\*\*

\* @deprecated 2020-07-08 no longer populated.

\*/

119: optional SupplementalLanguage supplemental\_language

// field 120, additional\_media\_metadata, is deprecated.

// field 121, media\_metadatas, is deprecated

// under certain circumstances, including long form tweets, we create and store a self-permalink

// to this tweet. in the case of a long-form tweet, this will be used in a truncated version

// of the tweet text.

122: optional ShortenedUrl self\_permalink

// metadata that is present on extended tweets.

123: optional ExtendedTweetMetadata extended\_tweet\_metadata

// obsolete 124: crosspost\_destinations.CrosspostDestinations crosspost\_destinations

// Communities associated with a tweet

125: optional Communities communities (personalDataType = 'PrivateTweetEntitiesAndMetadata', tweetEditAllowed='false')

// If some text at the beginning or end of the tweet should be hidden, then this

// field indicates the range of text that should be shown in clients.

126: optional TextRange visible\_text\_range

// @deprecated 2021-07-19

127: optional safety\_label.SafetyLabel spam\_high\_recall\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

128: optional safety\_label.SafetyLabel duplicate\_content\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

129: optional safety\_label.SafetyLabel live\_low\_quality\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

130: optional safety\_label.SafetyLabel nsfa\_high\_recall\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

131: optional safety\_label.SafetyLabel pdna\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

132: optional safety\_label.SafetyLabel search\_blacklist\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

133: optional safety\_label.SafetyLabel low\_quality\_mention\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

134: optional safety\_label.SafetyLabel bystander\_abusive\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

135: optional safety\_label.SafetyLabel automation\_high\_recall\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

136: optional safety\_label.SafetyLabel gore\_and\_violence\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

137: optional safety\_label.SafetyLabel untrusted\_url\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

138: optional safety\_label.SafetyLabel gore\_and\_violence\_high\_recall\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

139: optional safety\_label.SafetyLabel nsfw\_video\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

140: optional safety\_label.SafetyLabel nsfw\_near\_perfect\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

141: optional safety\_label.SafetyLabel automation\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

142: optional safety\_label.SafetyLabel nsfw\_card\_image\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

143: optional safety\_label.SafetyLabel duplicate\_mention\_label (personalDataType = 'TweetSafetyLabels')

// @deprecated 2021-07-19

144: optional safety\_label.SafetyLabel bounce\_label (personalDataType = 'TweetSafetyLabels')

// field 145 to 150 is reserved for safety labels

/\*\*

\* If this tweet is part of a self\_thread (tweetstorm) then this value may be set.

\* See SelfThreadMetadata for details.

\*/

151: optional SelfThreadMetadata self\_thread\_metadata

// field 152 has been deprecated

// The composer used to create this tweet. Either via the standard tweet creator or the

// Camera flow (go/newscamera).

//

// NOTE: this field is only set if a client passed an explicit ComposerSource in the PostTweetRequest.

// News Camera is deprecated and we no longer set ComposerSource in the PostTweetRequest so no new Tweets will

// have this field.

153: optional ComposerSource composer\_source

// Present if replies are restricted, see ConversationControl for more details

154: optional ConversationControl conversation\_control

// Determines the super follows requirements for being able to view a tweet.

155: optional ExclusiveTweetControl exclusive\_tweet\_control (tweetEditAllowed='false')

// Present for a Trusted Friends tweet, see TrustedFriendsControl for more details.

156: optional TrustedFriendsControl trusted\_friends\_control (tweetEditAllowed='false')

// Data about edits and editability. See EditControl for more details.

157: optional edit\_control.EditControl edit\_control

// Present for a CollabTweet or CollabInvitation, see CollabControl for more details.

158: optional CollabControl collab\_control (tweetEditAllowed='false')

// Present for a 3rd-party developer-built card. See http://go/developer-built-cards-prd

159: optional i64 developer\_built\_card\_id (personalDataType = 'CardId')

// Data about enrichments attached to a tweet.

160: optional creative\_entity\_enrichments.CreativeEntityEnrichments creative\_entity\_enrichments\_for\_tweet

// This field includes summed engagements from the previous tweets in the edit chain.

161: optional StatusCounts previous\_counts

// A list of media references, including information about the source Tweet for pasted media.

// Prefer this field to media\_keys, as media\_keys is not present for old Tweets or pasted media Tweets.

162: optional list<media\_ref.MediaRef> media\_refs

// Whether this tweet is a 'backend tweet' to be referenced only by the creatives containers service

// go/cea-cc-integration for more details

163: optional bool is\_creatives\_container\_backend\_tweet

/\*\*

\* Aggregated perspective of this tweet and all other versions from the point of view of the

\* user specified in for\_user\_id.

\*

\* This field is included only when for\_user\_id is provided and can be missing in a PARTIAL response

\* if the timeline service request fails.

\*/

164: optional api\_fields.TweetPerspective edit\_perspective

// Visibility controls related to Toxic Reply Filtering

// go/toxrf for more details

165: optional filtered\_reply\_details.FilteredReplyDetails filtered\_reply\_details

// The list of mentions that have unmentioned from the tweet's associated conversation

166: optional unmentions.UnmentionData unmention\_data

/\*\*

\* A list of users that were mentioned in the tweet and have a blocking

\* relationship with the author.

\*/

167: optional BlockingUnmentions blocking\_unmentions

/\*\*

\* A list of users that were mentioned in the tweet and should be unmentioned

\* based on their mention setttings

\*/

168: optional SettingsUnmentions settings\_unmentions

/\*\*

\* A Note associated with this Tweet.

\*/

169: optional note\_tweet.NoteTweet note\_tweet

// For additional fields, the next available field id is 169.

// NOTE: when adding a new additional field, please also update UnrequestedFieldScrubber.scrubKnownFields

/\*\*

\* INTERNAL FIELDS

\*

\* These fields are used by tweetypie only and should not be accessed externally.

\* The field ids are in descending order, starting with `32767`.

\*/

/\*\*

\* Present if tweet data is provided creatives container service instead of tweetypie storage,

\* with encapsulated tweets or customized data.

\*/

32763: optional i64 underlying\_creatives\_container\_id

/\*\*

\* Stores tweetypie-internal metadata about a DirectedAtUser.

\*

\* A tweet's DirectedAtUser is hydrated as follows:

\* 1. if this field is present, then DirectedAtUserMetadata.userId is the directed-at user

\* 2. if this field is absent, then if the tweet has a reply and has a mention starting at text

\* index 0 then that user is the directed-at user.

\*

\* Note: External clients should use CoreData.directed\_at\_user.

\*/

32764: optional DirectedAtUserMetadata directed\_at\_user\_metadata

// list of takedowns that are applied directly to the tweet

32765: optional list<withholding.TakedownReason> tweetypie\_only\_takedown\_reasons

// Stores the media keys used to interact with the media platform systems.

// Prefer `media\_refs` which will always have media data, unlike this field which is empty for

// older Tweets and Tweets with pasted media.

32766: optional list<MediaCommon.MediaKey> media\_keys

// field 32767 is the list of takedowns that are applied directly to the tweet

32767: optional list<string> tweetypie\_only\_takedown\_country\_codes (personalDataType = 'ContentRestrictionStatus')

// for internal fields, the next available field id is 32765 (counting down)

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