namespace java com.twitter.unified\_user\_actions.thriftjava

#@namespace scala com.twitter.unified\_user\_actions.thriftscala

#@namespace strato com.twitter.unified\_user\_actions

include "com/twitter/unified\_user\_actions/action\_info.thrift"

include "com/twitter/clientapp/gen/client\_app.thrift"

/\*

\* Tweet item information. Some development notes:

\* 1. Please keep this top-level struct as minimal as possible to reduce overhead.

\* 2. We intentionally avoid nesting action tweet in a separate structure

\* to underscore its importance and faciliate extraction of most commonly

\* needed fields such as actionTweetId. New fields related to the action tweet

\* should generally be prefixed with "actionTweet".

\* 3. For the related Tweets, e.g. retweetingTweetId, inReplyToTweetId, etc, we

\* mostly only keep their ids for consistency and simplicity.

\*/

struct TweetInfo {

/\* Id for the tweet that was actioned on \*/

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

// Deprecated, please don't re-use!

// 2: optional i64 actionTweetAuthorId(personalDataType = 'UserId')

/\* The social proof (i.e. banner) Topic Id that the action Tweet is associated to \*/

3: optional i64 actionTweetTopicSocialProofId(personalDataType='InferredInterests, ProvidedInterests')

4: optional AuthorInfo actionTweetAuthorInfo

// Fields 1-99 reserved for `actionFooBar` fields

/\* Additional details for the action that took place on actionTweetId \*/

100: optional action\_info.TweetActionInfo tweetActionInfo

/\* Id of the tweet retweeting the action tweet \*/

101: optional i64 retweetingTweetId(personalDataType = 'TweetId')

/\* Id of the tweet quoting the action Tweet, when the action type is quote \*/

102: optional i64 quotingTweetId(personalDataType = 'TweetId')

/\* Id of the tweet replying to the action Tweet, when the action type is reply \*/

103: optional i64 replyingTweetId(personalDataType = 'TweetId')

/\* Id of the tweet being quoted by the action tweet \*/

104: optional i64 quotedTweetId(personalDataType = 'TweetId')

/\* Id of the tweet being replied to by the action tweet \*/

105: optional i64 inReplyToTweetId(personalDataType = 'TweetId')

/\* Id of the tweet being retweeted by the action tweet, this is just for Unretweet action \*/

106: optional i64 retweetedTweetId(personalDataType = 'TweetId')

/\* Id of the tweet being edited, this is only available for TweetEdit action, and TweetDelete

\* action when the deleted tweet was created from Edit. \*/

107: optional i64 editedTweetId(personalDataType = 'TweetId')

/\* Position of a tweet item in a page such as home and tweet detail, and is populated in

\* Client Event. \*/

108: optional i32 tweetPosition

/\* PromotedId is provided by ads team for each promoted tweet and is logged in client event \*/

109: optional string promotedId(personalDataType = 'AdsId')

/\* corresponding to inReplyToTweetId \*/

110: optional i64 inReplyToAuthorId(personalDataType = 'UserId')

/\* corresponding to retweetingTweetId \*/

111: optional i64 retweetingAuthorId(personalDataType = 'UserId')

/\* corresponding to quotedTweetId \*/

112: optional i64 quotedAuthorId(personalDataType = 'UserId')

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

/\*

\* Profile item information. This follows TweetInfo's development notes.

\*/

struct ProfileInfo {

/\* Id for the profile (user\_id) that was actioned on

\*

\* In a social graph user action, e.g., user1 follows/blocks/mutes user2,

\* userIdentifier captures userId of user1 and actionProfileId records

\* the userId of user2.

\*/

1: required i64 actionProfileId(personalDataType = 'UserId')

// Fields 1-99 reserved for `actionFooBar` fields

/\* the full name of the user. max length is 50. \*/

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

/\* The handle/screenName of the user. This can't be changed.

\*/

3: optional string handle(personalDataType = 'UserName')

/\* the "bio" of the user. max length is 160. May contain one or more t.co

\* links, which will be hydrated in the UrlEntities substruct if the

\* QueryFields.URL\_ENTITIES is specified.

\*/

4: optional string description(personalDataType = 'Bio')

/\* Additional details for the action that took place on actionProfileId \*/

100: optional action\_info.ProfileActionInfo profileActionInfo

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

/\*

\* Topic item information. This follows TweetInfo's development notes.

\*/

struct TopicInfo {

/\* Id for the Topic that was actioned on \*/

1: required i64 actionTopicId(personalDataType='InferredInterests, ProvidedInterests')

// Fields 1-99 reserved for `actionFooBar` fields

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

/\*

\* Notification Item information.

\*

\* See go/phab-d973370-discuss, go/phab-d968144-discuss, and go/uua-action-type for details about

\* the schema design for Notification events.

\*/

struct NotificationInfo {

/\*

\* Id of the Notification was actioned on.

\*

\* Note that this field represents the `impressionId` of a Notification. It has been renamed to

\* `notificationId` in UUA so that the name effectively represents the value it holds,

\* i.e. a unique id for a Notification and request.

\*/

1: required string actionNotificationId(personalDataType='UniversallyUniqueIdentifierUuid')

/\*

\* Additional information contained in a Notification. This is a `union` arm to differentiate

\* among different types of Notifications and store relevant metadata for each type.

\*

\* For example, a Notification with a single Tweet will hold the Tweet id in `TweetNotification`.

\* Similarly, `MultiTweetNotification` is defined for Notiifcations with multiple Tweet ids.

\*

\* Refer to the definition of `union NotificationContent` below for more details.

\*/

2: required NotificationContent content

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

/\*

\* Additional information contained in a Notification.

\*/

union NotificationContent {

1: TweetNotification tweetNotification

2: MultiTweetNotification multiTweetNotification

// 3 - 100 reserved for other specific Notification types (for example, profile, event, etc.).

/\*

\* If a Notification cannot be categorized into any of the types at indices 1 - 100,

\* it is considered of `Unknown` type.

\*/

101: UnknownNotification unknownNotification

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

/\*

\* Notification contains exactly one `tweetId`.

\*/

struct TweetNotification {

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

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

/\*

\* Notification contains multiple `tweetIds`.

\* For example, user A receives a Notification when user B likes multiple Tweets authored by user A.

\*/

struct MultiTweetNotification {

1: required list<i64> tweetIds(personalDataType = 'TweetId')

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

/\*

\* Notification could not be categrized into known types at indices 1 - 100 in `NotificationContent`.

\*/

struct UnknownNotification {

// this field is just a placeholder since Sparrow doesn't support empty struct

100: optional bool placeholder

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

/\*

\* Trend Item information for promoted and non-promoted Trends.

\*/

struct TrendInfo {

/\*

\* Identifier for promoted Trends only.

\* This is not available for non-promoted Trends and the default value should be set to 0.

\*/

1: required i32 actionTrendId(personalDataType= 'TrendId')

/\*

\* Empty for promoted Trends only.

\* This should be set for all non-promoted Trends.

\*/

2: optional string actionTrendName

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

struct TypeaheadInfo {

/\* search query string \*/

1: required string actionQuery(personalDataType = 'SearchQuery')

2: required TypeaheadActionInfo typeaheadActionInfo

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

union TypeaheadActionInfo {

1: UserResult userResult

2: TopicQueryResult topicQueryResult

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

struct UserResult {

/\* The userId of the profile suggested in the typeahead drop-down, upon which the user took the action \*/

1: required i64 profileId(personalDataType = 'UserId')

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

struct TopicQueryResult {

/\* The topic query name suggested in the typeahead drop-down, upon which the user took the action \*/

1: required string suggestedTopicQuery(personalDataType = 'SearchQuery')

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

/\*

\* Item that captures feedback related information submitted by the user across modules / item (Eg: Search Results / Tweets)

\* Design discussion doc: https://docs.google.com/document/d/1UHiCrGzfiXOSymRAUM565KchVLZBAByMwvP4ARxeixY/edit#

\*/

struct FeedbackPromptInfo {

1: required FeedbackPromptActionInfo feedbackPromptActionInfo

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

union FeedbackPromptActionInfo {

1: DidYouFindItSearch didYouFindItSearch

2: TweetRelevantToSearch tweetRelevantToSearch

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

struct DidYouFindItSearch {

1: required string searchQuery(personalDataType= 'SearchQuery')

2: optional bool isRelevant

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

struct TweetRelevantToSearch {

1: required string searchQuery(personalDataType= 'SearchQuery')

2: required i64 tweetId

3: optional bool isRelevant

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

/\*

\* For (Tweet) Author info

\*/

struct AuthorInfo {

/\* In practice, this should be set. Rarely, it may be unset. \*/

1: optional i64 authorId(personalDataType = 'UserId')

/\* i.e. in-network (true) or out-of-network (false) \*/

2: optional bool isFollowedByActingUser

/\* i.e. is a follower (true) or not (false) \*/

3: optional bool isFollowingActingUser

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

/\*

\* Use for Call to Action events.

\*/

struct CTAInfo {

// this field is just a placeholder since Sparrow doesn't support empty struct

100: optional bool placeholder

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

/\*

\* Card Info

\*/

struct CardInfo {

1: optional i64 id

2: optional client\_app.ItemType itemType

// authorId is deprecated, please use AuthorInfo instead

// 3: optional i64 authorId(personalDataType = 'UserId')

4: optional AuthorInfo actionTweetAuthorInfo

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

/\*

\* When the user exits the app, the time (in millis) spent by them on the platform is recorded as User Active Seconds (UAS).

\*/

struct UASInfo {

1: required i64 timeSpentMs

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

/\*

\* Corresponding item for a user action.

\* An item should be treated independently if it has different affordances

\* (https://www.interaction-design.org/literature/topics/affordances) for the user.

\* For example, a Notification has different affordances than a Tweet in the Notification Tab;

\* in the former, you can either "click" or "see less often" and in the latter,

\* you can perform inline engagements such as "like" or "reply".

\* Note that an item may be rendered differently in different contexts, but as long as the

\* affordances remain the same or nearly similar, it can be treated as the same item

\* (e.g. Tweets can be rendered in slightly different ways in embeds vs in the app).

\* Item types (e.g. Tweets, Notifications) and ActionTypes should be 1:1, and when an action can be

\* performed on multiple types of items, consider granular action types.

\* For example, a user can take the Click action on Tweets and Notifications, and we have

\* separate ActionTypes for Tweet Click and Notification Click. This makes it easier to identify all the

\* actions associated with a particular item.

\*/

union Item {

1: TweetInfo tweetInfo

2: ProfileInfo profileInfo

3: TopicInfo topicInfo

4: NotificationInfo notificationInfo

5: TrendInfo trendInfo

6: CTAInfo ctaInfo

7: FeedbackPromptInfo feedbackPromptInfo

8: TypeaheadInfo typeaheadInfo

9: UASInfo uasInfo

10: CardInfo cardInfo

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