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/clientapp/gen/client\_app.thrift"

include "com/twitter/reportflow/report\_flow\_logs.thrift"

include "com/twitter/socialgraph/social\_graph\_service\_write\_log.thrift"

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

/\*

\* ActionType is typically a three part enum consisting of

\* [Origin][Item Type][Action Name]

\*

\* [Origin] is usually "client" or "server" to indicate how the action was derived.

\*

\* [Item Type] is singular and refers to the shorthand version of the type of

\* Item (e.g. Tweet, Profile, Notification instead of TweetInfo, ProfileInfo, NotificationInfo)

\* the action occurred on. Action types and item types should be 1:1, and when an action can be

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

\*

\* [Action Name] is the descriptive name of the user action (e.g. favorite, render impression);

\* action names should correspond to UI actions / ML labels (which are typically based on user

\* behavior from UI actions)

\*

\* Below are guidelines around naming of action types:

\* a) When an action is coupled to a product surface, be concise in naming such that the

\* combination of item type and action name captures the user behavior for the action in the UI. For example,

\* for an open on a Notification in the PushNotification product surface that is parsed from client events,

\* consider ClientNotificationOpen because the item Notification and the action name Open concisely represent

\* the action, and the product surface PushNotification can be identified independently.

\*

\* b) It is OK to use generic names like Click if needed to distinguish from another action OR

\* it is the best way to characterize an action concisely without confusion.

\* For example, for ClientTweetClickReply, this refers to actually clicking on the Reply button but not

\* Replying, and it is OK to include Click. Another example is Click on a Tweet anywhere (other than the fav,

\* reply, etc. buttons), which leads to the TweetDetails page. Avoid generic action names like Click if

\* there is a more specific UI aspect to reference and Click is implied, e.g. ClientTweetReport is

\* preferred over ClientTweetClickReport and ClientTweetReportClick.

\*

\* c) Rely on versioning found in the origin when it is present for action names. For example,

\* a "V2Impression" is named as such because in behavioral client events, there is

\* a "v2Impress" field. See go/bce-v2impress for more details.

\*

\* d) There is a distinction between "UndoAction" and "Un{Action}" action types.

\* An "UndoAction" is fired when a user clicks on the explicit "Undo" button, after they perform an action

\* This "Undo" button is a UI element that may be temporary, e.g.,

\* - the user waited too long to click the button, the button disappears from the UI (e.g., Undo for Mute, Block)

\* - the button does not disappear due to timeout, but becomes unavailable after the user closes a tab

\* (e.g, Undo for NotInterestedIn, NotAboutTopic)

\* Examples:

- ClientProfileUndoMute: a user clicks the "Undo" button after muting a Profile

- ClientTweetUndoNotInterestedIn: a users clicks the "Undo" button

after clicking "Not interested in this Tweet" button in the caret menu of a Tweet

\* An "Un{Action}" is fired when a user reverses a previous action, not by explicitly clicking an "Undo" button,

\* but through some other action that allows them to revert.

\* Examples:

\* - ClientProfileUnmute: a user clicks the "Unmute" button from the caret menu of the Profile they previously muted

\* - ClientTweetUnfav: a user unlikes a tweet by clicking on like button again

\*

\* Examples: ServerTweetFav, ClientTweetRenderImpression, ClientNotificationSeeLessOften

\*

\* See go/uua-action-type for more details.

\*/

enum ActionType {

// 0 - 999 used for actions derived from Server-side sources (e.g. Timelineservice, Tweetypie)

// NOTE: Please match values for corresponding server / client enum members (with offset 1000).

ServerTweetFav = 0

ServerTweetUnfav = 1

// Reserve 2 and 3 for ServerTweetLingerImpression and ServerTweetRenderImpression

ServerTweetCreate = 4

ServerTweetReply = 5

ServerTweetQuote = 6

ServerTweetRetweet = 7

// skip 8-10 since there are no server equivalents for ClickCreate, ClickReply, ClickQuote

// reserve 11-16 for server video engagements

ServerTweetDelete = 17 // User deletes a default tweet

ServerTweetUnreply = 18 // User deletes a reply tweet

ServerTweetUnquote = 19 // User deletes a quote tweet

ServerTweetUnretweet = 20 // User removes an existing retweet

// User edits a tweet. Edit will create a new tweet with editedTweetId = id of the original tweet

// The original tweet or the new tweet from edit can only be a default or quote tweet.

// A user can edit a default tweet to become a quote tweet (by adding the link to another Tweet),

// or edit a quote tweet to remove the quote and make it a default tweet.

// Both the initial tweet and the new tweet created from the edit can be edited, and each time the

// new edit will create a new tweet. All subsequent edits would have the same initial tweet id

// as the TweetInfo.editedTweetId.

// e.g. create Tweet A, edit Tweet A -> Tweet B, edit Tweet B -> Tweet C

// initial tweet id for both Tweet B anc Tweet C would be Tweet A

ServerTweetEdit = 21

// skip 22 for delete an edit if we want to add it in the future

// reserve 30-40 for server topic actions

// 41-70 reserved for all negative engagements and the related positive engagements

// For example, Follow and Unfollow, Mute and Unmute

// This is fired when a user click "Submit" at the end of a "Report Tweet" flow

// ClientTweetReport = 1041 is scribed by HealthClient team, on the client side

// This is scribed by spamacaw, on the server side

// They can be joined on reportFlowId

// See https://confluence.twitter.biz/pages/viewpage.action?spaceKey=HEALTH&title=Understanding+ReportDetails

ServerTweetReport = 41

// reserve 42 for ServerTweetNotInterestedIn

// reserve 43 for ServerTweetUndoNotInterestedIn

// reserve 44 for ServerTweetNotAboutTopic

// reserve 45 for ServerTweetUndoNotAboutTopic

ServerProfileFollow = 50 // User follows a Profile

ServerProfileUnfollow = 51 // User unfollows a Profile

ServerProfileBlock = 52 // User blocks a Profile

ServerProfileUnblock = 53 // User unblocks a Profile

ServerProfileMute = 54 // User mutes a Profile

ServerProfileUnmute = 55 // User unmutes a Profile

// User reports a Profile as Spam / Abuse

// This user action type includes ProfileReportAsSpam and ProfileReportAsAbuse

ServerProfileReport = 56

// reserve 57 for ServerProfileUnReport

// reserve 56-70 for server social graph actions

// 71-90 reserved for click-based events

// reserve 71 for ServerTweetClick

// 1000 - 1999 used for actions derived from Client-side sources (e.g. Client Events, BCE)

// NOTE: Please match values for corresponding server / client enum members (with offset 1000).

// 1000 - 1499 used for legacy client events

ClientTweetFav = 1000

ClientTweetUnfav = 1001

ClientTweetLingerImpression = 1002

// Please note that: Render impression for quoted Tweets would emit 2 events:

// 1 for the quoting Tweet and 1 for the original Tweet!!!

ClientTweetRenderImpression = 1003

// 1004 reserved for ClientTweetCreate

// This is "Send Reply" event to indicate publishing of a reply Tweet as opposed to clicking

// on the reply button to initiate a reply Tweet (captured in ClientTweetClickReply).

// The differences between this and the ServerTweetReply are:

// 1) ServerTweetReply already has the new Tweet Id 2) A sent reply may be lost during transfer

// over the wire and thus may not end up with a follow-up ServerTweetReply.

ClientTweetReply = 1005

// This is the "send quote" event to indicate publishing of a quote tweet as opposed to clicking

// on the quote button to initiate a quote tweet (captured in ClientTweetClickQuote).

// The differences between this and the ServerTweetQuote are:

// 1) ServerTweetQuote already has the new Tweet Id 2) A sent quote may be lost during transfer

// over the wire and thus may not end up with a follow-up ServerTweetQuote.

ClientTweetQuote = 1006

// This is the "retweet" event to indicate publishing of a retweet.

ClientTweetRetweet = 1007

// 1008 reserved for ClientTweetClickCreate

// This is user clicking on the Reply button not actually sending a reply Tweet,

// thus the name ClickReply

ClientTweetClickReply = 1009

// This is user clicking the Quote/RetweetWithComment button not actually sending the quote,

// thus the name ClickQuote

ClientTweetClickQuote = 1010

// 1011 - 1016: Refer to go/cme-scribing and go/interaction-event-spec for details

// This is fired when playback reaches 25% of total track duration. Not valid for live videos.

// For looping playback, this is only fired once and does not reset at loop boundaries.

ClientTweetVideoPlayback25 = 1011

// This is fired when playback reaches 50% of total track duration. Not valid for live videos.

// For looping playback, this is only fired once and does not reset at loop boundaries.

ClientTweetVideoPlayback50 = 1012

// This is fired when playback reaches 75% of total track duration. Not valid for live videos.

// For looping playback, this is only fired once and does not reset at loop boundaries.

ClientTweetVideoPlayback75 = 1013

// This is fired when playback reaches 95% of total track duration. Not valid for live videos.

// For looping playback, this is only fired once and does not reset at loop boundaries.

ClientTweetVideoPlayback95 = 1014

// This if fired when the video has been played in non-preview

// (i.e. not autoplaying in the timeline) mode, and was not started via auto-advance.

// For looping playback, this is only fired once and does not reset at loop boundaries.

ClientTweetVideoPlayFromTap = 1015

// This is fired when 50% of the video has been on-screen and playing for 10 consecutive seconds

// or 95% of the video duration, whichever comes first.

// For looping playback, this is only fired once and does not reset at loop boundaries.

ClientTweetVideoQualityView = 1016

// Fired when either view\_threshold or play\_from\_tap is fired.

// For looping playback, this is only fired once and does not reset at loop boundaries.

ClientTweetVideoView = 1109

// Fired when 50% of the video has been on-screen and playing for 2 consecutive seconds,

// regardless of video duration.

// For looping playback, this is only fired once and does not reset at loop boundaries.

ClientTweetVideoMrcView = 1110

// Fired when the video is:

// - Playing for 3 cumulative (not necessarily consecutive) seconds with 100% in view for looping video.

// - Playing for 3 cumulative (not necessarily consecutive) seconds or the video duration, whichever comes first, with 100% in view for non-looping video.

// For looping playback, this is only fired once and does not reset at loop boundaries.

ClientTweetVideoViewThreshold = 1111

// Fired when the user clicks a generic ‘visit url’ call to action.

ClientTweetVideoCtaUrlClick = 1112

// Fired when the user clicks a ‘watch now’ call to action.

ClientTweetVideoCtaWatchClick = 1113

// 1017 reserved for ClientTweetDelete

// 1018-1019 for Client delete a reply and delete a quote if we want to add them in the future

// This is fired when a user clicks on "Undo retweet" after re-tweeting a tweet

ClientTweetUnretweet = 1020

// 1021 reserved for ClientTweetEdit

// 1022 reserved for Client delete an edit if we want to add it in the future

// This is fired when a user clicks on a photo within a tweet and the photo expands to fit

// the screen.

ClientTweetPhotoExpand = 1023

// This is fired when a user clicks on a profile mention inside a tweet.

ClientTweetClickMentionScreenName = 1024

// 1030 - 1035 for topic actions

// There are multiple cases:

// 1. Follow from the Topic page (or so-called landing page)

// 2. Click on Tweet's caret menu of "Follow (the topic)", it needs to be:

// 1) user follows the Topic already (otherwise there is no "Follow" menu by default),

// 2) and clicked on the "Unfollow Topic" first.

ClientTopicFollow = 1030

// There are multiple cases:

// 1. Unfollow from the Topic page (or so-called landing page)

// 2. Click on Tweet's caret menu of "Unfollow (the topic)" if the user has already followed

// the topic.

ClientTopicUnfollow = 1031

// This is fired when the user clicks the "x" icon next to the topic on their timeline,

// and clicks "Not interested in {TOPIC}" in the pop-up prompt

// Alternatively, they can also click "See more" button to visit the topic page, and click "Not interested" there.

ClientTopicNotInterestedIn = 1032

// This is fired when the user clicks the "Undo" button after clicking "x" or "Not interested" on a Topic

// which is captured in ClientTopicNotInterestedIn

ClientTopicUndoNotInterestedIn = 1033

// 1036-1070 reserved for all negative engagements and the related positive engagements

// For example, Follow and Unfollow, Mute and Unmute

// This is fired when a user clicks on "This Tweet's not helpful" flow in the caret menu

// of a Tweet result on the Search Results Page

ClientTweetNotHelpful = 1036

// This is fired when a user clicks Undo after clicking on

// "This Tweet's not helpful" flow in the caret menu of a Tweet result on the Search Results Page

ClientTweetUndoNotHelpful = 1037

// This is fired when a user starts and/or completes the "Report Tweet" flow in the caret menu of a Tweet

ClientTweetReport = 1041

/\*

\* 1042-1045 refers to actions that are related to the

\* "Not Interested In" button in the caret menu of a Tweet.

\*

\* ClientTweetNotInterestedIn is fired when a user clicks the

\* "Not interested in this Tweet" button in the caret menu of a Tweet.

\* A user can undo the ClientTweetNotInterestedIn action by clicking the

\* "Undo" button that appears as a prompt in the caret menu, resulting

\* in ClientTweetUndoNotInterestedIn being fired.

\* If a user chooses to not undo and proceed, they are given multiple choices

\* in a prompt to better document why they are not interested in a Tweet.

\* For example, if a Tweet is not about a Topic, a user can click

\* "This Tweet is not about {TOPIC}" in the provided prompt, resulting in

\* in ClientTweetNotAboutTopic being fired.

\* A user can undo the ClientTweetNotAboutTopic action by clicking the "Undo"

\* button that appears as a subsequent prompt in the caret menu. Undoing this action

\* results in the previous UI state, where the user had only marked "Not Interested In" and

\* can still undo the original ClientTweetNotInterestedIn action.

\* Similarly a user can select "This Tweet isn't recent" action resulting in ClientTweetNotRecent

\* and he could undo this action immediately which results in ClientTweetUndoNotRecent

\* Similarly a user can select "Show fewer tweets from" action resulting in ClientTweetSeeFewer

\* and he could undo this action immediately which results in ClientTweetUndoSeeFewer

\*/

ClientTweetNotInterestedIn = 1042

ClientTweetUndoNotInterestedIn = 1043

ClientTweetNotAboutTopic = 1044

ClientTweetUndoNotAboutTopic = 1045

ClientTweetNotRecent = 1046

ClientTweetUndoNotRecent = 1047

ClientTweetSeeFewer = 1048

ClientTweetUndoSeeFewer = 1049

// This is fired when a user follows a profile from the

// profile page header / people module and people tab on the Search Results Page / sidebar on the Home page

// A Profile can also be followed when a user clicks follow in the caret menu of a Tweet

// or follow button on hovering on profile avatar, which is captured in ClientTweetFollowAuthor = 1060

ClientProfileFollow = 1050

// reserve 1050/1051 for client side Follow/Unfollow

// This is fired when a user clicks Block in a Profile page

// A Profile can also be blocked when a user clicks Block in the caret menu of a Tweet,

// which is captured in ClientTweetBlockAuthor = 1062

ClientProfileBlock = 1052

// This is fired when a user clicks unblock in a pop-up prompt right after blocking a profile

// in the profile page or clicks unblock in a drop-down menu in the profile page.

ClientProfileUnblock = 1053

// This is fired when a user clicks Mute in a Profile page

// A Profile can also be muted when a user clicks Mute in the caret menu of a Tweet, which is captured in ClientTweetMuteAuthor = 1064

ClientProfileMute = 1054

// reserve 1055 for client side Unmute

// This is fired when a user clicks "Report User" action from user profile page

ClientProfileReport = 1056

// reserve 1057 for ClientProfileUnreport

// This is fired when a user clicks on a profile from all modules except tweets

// (eg: People Search / people module in Top tab in Search Result Page

// For tweets, the click is captured in ClientTweetClickProfile

ClientProfileClick = 1058

// reserve 1059-1070 for client social graph actions

// This is fired when a user clicks Follow in the caret menu of a Tweet or hovers on the avatar of the tweet

// author and clicks on the Follow button. A profile can also be followed by clicking the Follow button on the

// Profile page and confirm, which is captured in ClientProfileFollow. The event emits two items, one of user type

// and another of tweet type, since the default implementation of BaseClientEvent only looks for Tweet type,

// the other item is dropped which is the expected behaviour

ClientTweetFollowAuthor = 1060

// This is fired when a user clicks Unfollow in the caret menu of a Tweet or hovers on the avatar of the tweet

// author and clicks on the Unfollow button. A profile can also be unfollowed by clicking the Unfollow button on the

// Profile page and confirm, which will be captured in ClientProfileUnfollow. The event emits two items, one of user type

// and another of tweet type, since the default implementation of BaseClientEvent only looks for Tweet type,

// the other item is dropped which is the expected behaviour

ClientTweetUnfollowAuthor = 1061

// This is fired when a user clicks Block in the menu of a Tweet to block the Profile that

// authored this Tweet. A Profile can also be blocked in the Profile page, which is captured

// in ClientProfileBlock = 1052

ClientTweetBlockAuthor = 1062

// This is fired when a user clicks unblock in a pop-up prompt right after blocking an author

// in the drop-down menu of a tweet

ClientTweetUnblockAuthor = 1063

// This is fired when a user clicks Mute in the menu of a Tweet to block the Profile that

// authored this Tweet. A Profile can also be muted in the Profile page, which is captured in ClientProfileMute = 1054

ClientTweetMuteAuthor = 1064

// reserve 1065 for ClientTweetUnmuteAuthor

// 1071-1090 reserved for click-based events

// click-based events are defined as clicks on a UI container (e.g., tweet, profile, etc.), as opposed to clearly named

// button or menu (e.g., follow, block, report, etc.), which requires a specific action name than "click".

// This is fired when a user clicks on a Tweet to open the Tweet details page. Note that for

// Tweets in the Notification Tab product surface, a click can be registered differently

// depending on whether the Tweet is a rendered Tweet (a click results in ClientTweetClick)

// or a wrapper Notification (a click results in ClientNotificationClick).

ClientTweetClick = 1071

// This is fired when a user clicks to view the profile page of a user from a tweet

// Contains a TweetInfo of this tweet

ClientTweetClickProfile = 1072

// This is fired when a user clicks on the "share" icon on a Tweet to open the share menu.

// The user may or may not proceed and finish sharing the Tweet.

ClientTweetClickShare = 1073

// This is fired when a user clicks "Copy link to Tweet" in a menu appeared after hitting

// the "share" icon on a Tweet OR when a user selects share\_via -> copy\_link after long-click

// a link inside a tweet on a mobile device

ClientTweetShareViaCopyLink = 1074

// This is fired when a user clicks "Send via Direct Message" after

// clicking on the "share" icon on a Tweet to open the share menu.

// The user may or may not proceed and finish Sending the DM.

ClientTweetClickSendViaDirectMessage = 1075

// This is fired when a user clicks "Bookmark" after

// clicking on the "share" icon on a Tweet to open the share menu.

ClientTweetShareViaBookmark = 1076

// This is fired when a user clicks "Remove Tweet from Bookmarks" after

// clicking on the "share" icon on a Tweet to open the share menu.

ClientTweetUnbookmark = 1077

// This is fired when a user clicks on the hashtag in a Tweet.

// The click on hashtag in "What's happening" section gives you other scribe '\*:\*:sidebar:\*:trend:search'

// Currenly we are only filtering for itemType=Tweet. There are other items present in the event where itemType = user

// but those items are in dual-events (events with multiple itemTypes) and happen when you click on a hashtag in a Tweet from someone's profile,

// hence we are ignoring those itemType and only keeping itemType=Tweet.

ClientTweetClickHashtag = 1078

// This is fired when a user clicks "Bookmark" after clicking on the "share" icon on a Tweet to open the share menu, or

// when a user clicks on the 'bookmark' icon on a Tweet (bookmark icon is available to ios only as of March 2023).

// TweetBookmark and TweetShareByBookmark log the same events but serve for individual use cases.

ClientTweetBookmark = 1079

// 1078 - 1089 for all Share related actions.

// This is fired when a user clicks on a link in a tweet.

// The link could be displayed as a URL or embedded in a component such as an image or a card in a tweet.

ClientTweetOpenLink = 1090

// This is fired when a user takes screenshot.

// This is available for mobile clients only.

ClientTweetTakeScreenshot = 1091

// 1100 - 1101: Refer to go/cme-scribing and go/interaction-event-spec for details

// Fired on the first tick of a track regardless of where in the video it is playing.

// For looping playback, this is only fired once and does not reset at loop boundaries.

ClientTweetVideoPlaybackStart = 1100

// Fired when playback reaches 100% of total track duration.

// Not valid for live videos.

// For looping playback, this is only fired once and does not reset at loop boundaries.

ClientTweetVideoPlaybackComplete = 1101

// A user can select "This Tweet isn't relevant" action resulting in ClientTweetNotRelevant

// and they could undo this action immediately which results in ClientTweetUndoNotRelevant

ClientTweetNotRelevant = 1102

ClientTweetUndoNotRelevant = 1103

// A generic action type to submit feedback for different modules / items ( Tweets / Search Results )

ClientFeedbackPromptSubmit = 1104

// This is fired when a user profile is open in a Profile page

ClientProfileShow = 1105

/\*

\* This is triggered when a user exits the Twitter platform. The amount of the time spent on the

\* platform is recorded in ms that can be used to compute the User Active Seconds (UAS).

\*/

ClientAppExit = 1106

/\*

\* For "card" related actions

\*/

ClientCardClick = 1107

ClientCardOpenApp = 1108

ClientCardAppInstallAttempt = 1114

ClientPollCardVote = 1115

/\*

\* The impressions 1121-1123 together with the ClientTweetRenderImpression 1003 are used by ViewCount

\* and UnifiedEngagementCounts as EngagementType.Displayed and EngagementType.Details.

\*

\* For definitions, please refer to https://sourcegraph.twitter.biz/git.twitter.biz/source/-/blob/common-internal/analytics/client-event-util/src/main/java/com/twitter/common\_internal/analytics/client\_event\_util/TweetImpressionUtils.java?L14&subtree=true

\*/

ClientTweetGalleryImpression = 1121

ClientTweetDetailsImpression = 1122

/\*\*

\* This is fired when a user is logged out and follows a profile from the

\* profile page / people module from web.

\* One can only try to follow from web because iOS and Android do not support logged out browsing as of Jan 2023.

\*/

ClientProfileFollowAttempt = 1200

/\*\*

\* This is fired when a user is logged out and favourite a tweet from web.

\* One can only try to favourite from web, iOS and Android do not support logged out browsing

\*/

ClientTweetFavoriteAttempt = 1201

/\*\*

\* This is fired when a user is logged out and Retweet a tweet from web.

\* One can only try to favourite from web, iOS and Android do not support logged out browsing

\*/

ClientTweetRetweetAttempt = 1202

/\*\*

\* This is fired when a user is logged out and reply on tweet from web.

\* One can only try to favourite from web, iOS and Android do not support logged out browsing

\*/

ClientTweetReplyAttempt = 1203

/\*\*

\* This is fired when a user is logged out and clicks on login button.

\* Currently seem to be generated only on [m5, LiteNativeWrapper]

\*/

ClientCTALoginClick = 1204

/\*\*

\* This is fired when a user is logged out and login window is shown.

\*/

ClientCTALoginStart = 1205

/\*\*

\* This is fired when a user is logged out and login is successful.

\*/

ClientCTALoginSuccess = 1206

/\*\*

\* This is fired when a user is logged out and clicks on signup button.

\*/

ClientCTASignupClick = 1207

/\*\*

\* This is fired when a user is logged out and signup is successful.

\*/

ClientCTASignupSuccess = 1208

// 1400 - 1499 for product surface specific actions

// This is fired when a user opens a Push Notification

ClientNotificationOpen = 1400

// This is fired when a user clicks on a Notification in the Notification Tab

ClientNotificationClick = 1401

// This is fired when a user taps the "See Less Often" caret menu item of a Notification in the Notification Tab

ClientNotificationSeeLessOften = 1402

// This is fired when a user closes or swipes away a Push Notification

ClientNotificationDismiss = 1403

// 1420 - 1439 is reserved for Search Results Page related actions

// 1440 - 1449 is reserved for Typeahead related actions

// This is fired when a user clicks on a typeahead suggestion(queries, events, topics, users)

// in a drop-down menu of a search box or a tweet compose box.

ClientTypeaheadClick = 1440

// 1500 - 1999 used for behavioral client events

// Tweet related impressions

ClientTweetV2Impression = 1500

/\* Fullscreen impressions

\*

\* Android client will always log fullscreen\_video impressions, regardless of the media type

\* i.e. video, image, MM will all be logged as fullscreen\_video

\*

\* iOS clients will log fullscreen\_video or fullscreen\_image depending on the media type

\* on display when the user exits fullscreen. i.e.

\* - image tweet => fullscreen\_image

\* - video tweet => fullscreen\_video

\* - MM tweet => fullscreen\_video if user exits fullscreen from the video

\* => fullscreen\_image if user exits fullscreen from the image

\*

\* Web clients will always log fullscreen\_image impressions, regardless of the media type

\*

\* References

\* https://docs.google.com/document/d/1oEt9\_Gtz34cmO\_JWNag5YKKEq4Q7cJFL-nbHOmhnq1Y

\* https://docs.google.com/document/d/1V\_7TbfPvTQgtE\_91r5SubD7n78JsVR\_iToW59gOMrfQ

\*/

ClientTweetVideoFullscreenV2Impression = 1501

ClientTweetImageFullscreenV2Impression = 1502

// Profile related impressions

ClientProfileV2Impression = 1600

/\*

\* Email Notifications: These are actions taken by the user in response to Your Highlights email

\* ClientTweetEmailClick refers to the action NotificationType.Click

\*/

ClientTweetEmailClick = 5001

/\*

\* User create via Gizmoduck

\*/

ServerUserCreate = 6000

ServerUserUpdate = 6001

/\*

\* Ads callback engagements

\*/

/\*

\* This engagement is generated when a user Favs a promoted Tweet.

\*/

ServerPromotedTweetFav = 7000

/\*

\* This engagement is generated when a user Unfavs a promoted Tweet that they previously Faved.

\*/

ServerPromotedTweetUnfav = 7001

ServerPromotedTweetReply = 7002

ServerPromotedTweetRetweet = 7004

/\*

\* The block could be performed from the promoted tweet or on the promoted tweet's author's profile

\* ads\_spend\_event data shows majority (~97%) of blocks have an associated promoted tweet id

\* So for now we assume the blocks are largely performed from the tweet and following naming convention of ClientTweetBlockAuthor

\*/

ServerPromotedTweetBlockAuthor = 7006

ServerPromotedTweetUnblockAuthor = 7007

/\*

\* This is when a user clicks on the Conversational Card in the Promoted Tweet which

\* leads to the Tweet Compose page. The user may or may not send the new Tweet.

\*/

ServerPromotedTweetComposeTweet = 7008

/\*

\* This is when a user clicks on the Promoted Tweet to view its details/replies.

\*/

ServerPromotedTweetClick = 7009

/\*

\* The video ads engagements are divided into two sets: VIDEO\_CONTENT\_\* and VIDEO\_AD\_\*. These engagements

\* have similar definitions. VIDEO\_CONTENT\_\* engagements are fired for videos that are part of

\* a Tweet. VIDEO\_AD\_\* engagements are fired for a preroll ad. A preroll ad can play on a promoted

\* Tweet or on an organic Tweet. go/preroll-matching for more information.

\*

\* 7011-7013: A Promoted Event is fired when playback reaches 25%, 50%, 75% of total track duration.

\* This is for the video on a promoted Tweet.

\* Not valid for live videos. Refer go/avscribing.

\* For a video that has a preroll ad played before it, the metadata will contain information about

\* the preroll ad as well as the video itself. There will be no preroll metadata if there was no

\* preroll ad played.

\*/

ServerPromotedTweetVideoPlayback25 = 7011

ServerPromotedTweetVideoPlayback50 = 7012

ServerPromotedTweetVideoPlayback75 = 7013

/\*

\* This is when a user successfully completes the Report flow on a Promoted Tweet.

\* It covers reports for all policies from Client Event.

\*/

ServerPromotedTweetReport = 7041

/\*

\* Follow from Ads data stream, it could be from both Tweet or other places

\*/

ServerPromotedProfileFollow = 7060

/\*

\* Follow from Ads data stream, it could be from both Tweet or other places

\*/

ServerPromotedProfileUnfollow = 7061

/\*

\* This is when a user clicks on the mute promoted tweet's author option from the menu.

\*/

ServerPromotedTweetMuteAuthor = 7064

/\*

\* This is fired when a user clicks on the profile image, screen name, or the user name of the

\* author of the Promoted Tweet which leads to the author's profile page.

\*/

ServerPromotedTweetClickProfile = 7072

/\*

\* This is fired when a user clicks on a hashtag in the Promoted Tweet.

\*/

ServerPromotedTweetClickHashtag = 7078

/\*

\* This is fired when a user opens link by clicking on a URL in the Promoted Tweet.

\*/

ServerPromotedTweetOpenLink = 7079

/\*

\* This is fired when a user swipes to the next element of the carousel in the Promoted Tweet.

\*/

ServerPromotedTweetCarouselSwipeNext = 7091

/\*

\* This is fired when a user swipes to the previous element of the carousel in the Promoted Tweet.

\*/

ServerPromotedTweetCarouselSwipePrevious = 7092

/\*

\* This event is only for the Promoted Tweets with a web URL.

\* It is fired after exiting a WebView from a Promoted Tweet if the user was on the WebView for

\* at least 1 second.

\*

\* See https://confluence.twitter.biz/display/REVENUE/dwell\_short for more details.

\*/

ServerPromotedTweetLingerImpressionShort = 7093

/\*

\* This event is only for the Promoted Tweets with a web URL.

\* It is fired after exiting a WebView from a Promoted Tweet if the user was on the WebView for

\* at least 2 seconds.

\*

\* See https://confluence.twitter.biz/display/REVENUE/dwell\_medium for more details.

\*/

ServerPromotedTweetLingerImpressionMedium = 7094

/\*

\* This event is only for the Promoted Tweets with a web URL.

\* It is fired after exiting a WebView from a Promoted Tweet if the user was on the WebView for

\* at least 10 seconds.

\*

\* See https://confluence.twitter.biz/display/REVENUE/dwell\_long for more details.

\*/

ServerPromotedTweetLingerImpressionLong = 7095

/\*

\* This is fired when a user navigates to explorer page (taps search magnifying glass on Home page)

\* and a Promoted Trend is present and taps ON the promoted spotlight - a video/gif/image in the

\* "hero" position (top of the explorer page).

\*/

ServerPromotedTweetClickSpotlight = 7096

/\*

\* This is fired when a user navigates to explorer page (taps search magnifying glass on Home page)

\* and a Promoted Trend is present.

\*/

ServerPromotedTweetViewSpotlight = 7097

/\*

\* 7098-7099: Promoted Trends appear in the first or second slots of the “Trends for you” section

\* in the Explore tab and “What’s Happening” module on Twitter.com. For more information, check go/ads-takeover.

\* 7099: This is fired when a user views a promoted Trend. It should be considered as an impression.

\*/

ServerPromotedTrendView = 7098

/\*

\* 7099: This is fired when a user clicks a promoted Trend. It should be considered as an engagment.

\*/

ServerPromotedTrendClick = 7099

/\*

\* 7131-7133: A Promoted Event fired when playback reaches 25%, 50%, 75% of total track duration.

\* This is for the preroll ad that plays before a video on a promoted Tweet.

\* Not valid for live videos. Refer go/avscribing.

\* This will only contain metadata for the preroll ad.

\*/

ServerPromotedTweetVideoAdPlayback25 = 7131

ServerPromotedTweetVideoAdPlayback50 = 7132

ServerPromotedTweetVideoAdPlayback75 = 7133

/\*

\* 7151-7153: A Promoted Event fired when playback reaches 25%, 50%, 75% of total track duration.

\* This is for the preroll ad that plays before a video on an organic Tweet.

\* Not valid for live videos. Refer go/avscribing.

\* This will only contain metadata for the preroll ad.

\*/

ServerTweetVideoAdPlayback25 = 7151

ServerTweetVideoAdPlayback50 = 7152

ServerTweetVideoAdPlayback75 = 7153

ServerPromotedTweetDismissWithoutReason = 7180

ServerPromotedTweetDismissUninteresting = 7181

ServerPromotedTweetDismissRepetitive = 7182

ServerPromotedTweetDismissSpam = 7183

/\*

\* For FavoriteArchival Events

\*/

ServerTweetArchiveFavorite = 8000

ServerTweetUnarchiveFavorite = 8001

/\*

\* For RetweetArchival Events

\*/

ServerTweetArchiveRetweet = 8002

ServerTweetUnarchiveRetweet = 8003

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

/\*

\* This union will be updated when we have a particular

\* action that has attributes unique to that particular action

\* (e.g. linger impressions have start/end times) and not common

\* to all tweet actions.

\* Naming convention for TweetActionInfo should be consistent with

\* ActionType. For example, `ClientTweetLingerImpression` ActionType enum

\* should correspond to `ClientTweetLingerImpression` TweetActionInfo union arm.

\* We typically preserve 1:1 mapping between ActionType and TweetActionInfo. However, we make

\* exceptions when optimizing for customer requirements. For example, multiple 'ClientTweetVideo\*'

\* ActionType enums correspond to a single `TweetVideoWatch` TweetActionInfo union arm because

\* customers want individual action labels but common information across those labels.

\*/

union TweetActionInfo {

// 41 matches enum index ServerTweetReport in ActionType

41: ServerTweetReport serverTweetReport

// 1002 matches enum index ClientTweetLingerImpression in ActionType

1002: ClientTweetLingerImpression clientTweetLingerImpression

// Common metadata for

// 1. "ClientTweetVideo\*" ActionTypes with enum indices 1011-1016 and 1100-1101

// 2. "ServerPromotedTweetVideo\*" ActionTypes with enum indices 7011-7013 and 7131-7133

// 3. "ServerTweetVideo\*" ActionTypes with enum indices 7151-7153

// This is because:

// 1. all the above listed ActionTypes share common metadata

// 2. more modular code as the same struct can be reused

// 3. reduces chance of error while populating and parsing the metadata

// 4. consumers can easily process the metadata

1011: TweetVideoWatch tweetVideoWatch

// 1012: skip

// 1013: skip

// 1014: skip

// 1015: skip

// 1016: skip

// 1024 matches enum index ClientTweetClickMentionScreenName in ActionType

1024: ClientTweetClickMentionScreenName clientTweetClickMentionScreenName

// 1041 matches enum index ClientTweetReport in ActionType

1041: ClientTweetReport clientTweetReport

// 1060 matches enum index ClientTweetFollowAuthor in ActionType

1060: ClientTweetFollowAuthor clientTweetFollowAuthor

// 1061 matches enum index ClientTweetUnfollowAuthor in ActionType

1061: ClientTweetUnfollowAuthor clientTweetUnfollowAuthor

// 1078 matches enum index ClientTweetClickHashtag in ActionType

1078: ClientTweetClickHashtag clientTweetClickHashtag

// 1090 matches enum index ClientTweetOpenLink in ActionType

1090: ClientTweetOpenLink clientTweetOpenLink

// 1091 matches enum index ClientTweetTakeScreenshot in ActionType

1091: ClientTweetTakeScreenshot clientTweetTakeScreenshot

// 1500 matches enum index ClientTweetV2Impression in ActionType

1500: ClientTweetV2Impression clientTweetV2Impression

// 7079 matches enum index ServerPromotedTweetOpenLink in ActionType

7079: ServerPromotedTweetOpenLink serverPromotedTweetOpenLink

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

struct ClientTweetOpenLink {

//Url which was clicked.

1: optional string url(personalDataType = 'RawUrlPath')

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

struct ServerPromotedTweetOpenLink {

//Url which was clicked.

1: optional string url(personalDataType = 'RawUrlPath')

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

struct ClientTweetClickHashtag {

/\* Hashtag string which was clicked. The PDP annotation is SearchQuery,

\* because clicking on the hashtag triggers a search with the hashtag

\*/

1: optional string hashtag(personalDataType = 'SearchQuery')

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

struct ClientTweetTakeScreenshot {

//percentage visible height.

1: optional i32 percentVisibleHeight100k

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

/\*

\* See go/ioslingerimpressionbehaviors and go/lingerandroidfaq

\* for ios and android client definitions of a linger respectively.

\*/

struct ClientTweetLingerImpression {

/\* Milliseconds since epoch when the tweet became more than 50% visible. \*/

1: required i64 lingerStartTimestampMs(personalDataType = 'ImpressionMetadata')

/\* Milliseconds since epoch when the tweet became less than 50% visible. \*/

2: required i64 lingerEndTimestampMs(personalDataType = 'ImpressionMetadata')

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

/\*

\* See go/behavioral-client-events for general behavioral client event (BCE) information

\* and go/bce-v2impress for detailed information about BCE impression events.

\*

\* Unlike ClientTweetLingerImpression, there is no lower bound on the amount of time

\* necessary for the impress event to occur. There is also no visibility requirement for a impress

\* event to occur.

\*/

struct ClientTweetV2Impression {

/\* Milliseconds since epoch when the tweet became visible. \*/

1: required i64 impressStartTimestampMs(personalDataType = 'ImpressionMetadata')

/\* Milliseconds since epoch when the tweet became visible. \*/

2: required i64 impressEndTimestampMs(personalDataType = 'ImpressionMetadata')

/\*

\* The UI component that hosted this tweet where the impress event happened.

\*

\* For example, sourceComponent = "tweet" if the impress event happened on a tweet displayed amongst

\* a collection of tweets, or sourceComponent = "tweet\_details" if the impress event happened on

\* a tweet detail UI component.

\*/

3: required string sourceComponent(personalDataType = 'WebsitePage')

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

/\*

\* Refer to go/cme-scribing and go/interaction-event-spec for details

\*/

struct TweetVideoWatch {

/\*

\* Type of video included in the Tweet

\*/

1: optional client\_app.MediaType mediaType(personalDataType = 'MediaFile')

/\*

\* Whether the video content is "monetizable", i.e.,

\* if a preroll ad may be served dynamically when the video plays

\*/

2: optional bool isMonetizable(personalDataType = 'MediaFile')

/\*

\* The owner of the video, provided by playlist.

\*

\* For ad engagements related to a preroll ad (VIDEO\_AD\_\*),

\* this will be the owner of the preroll ad and same as the prerollOwnerId.

\*

\* For ad engagements related to a regular video (VIDEO\_CONTENT\_\*), this will be the owner of the

\* video and not the preroll ad.

\*/

3: optional i64 videoOwnerId(personalDataType = 'UserId')

/\*

\* Identifies the video associated with a card.

\*

\* For ad Engagements, in the case of engagements related to a preroll ad (VIDEO\_AD\_\*),

\* this will be the id of the preroll ad and same as the prerollUuid.

\*

\* For ad engagements related to a regular video (VIDEO\_CONTENT\_\*), this will be id of the video

\* and not the preroll ad.

\*/

4: optional string videoUuid(personalDataType = 'MediaId')

/\*

\* Id of the preroll ad shown before the video

\*/

5: optional string prerollUuid(personalDataType = 'MediaId')

/\*

\* Advertiser id of the preroll ad

\*/

6: optional i64 prerollOwnerId(personalDataType = 'UserId')

/\*

\* for amplify\_flayer events, indicates whether preroll or the main video is being played

\*/

7: optional string videoType(personalDataType = 'MediaFile')

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

struct ClientTweetClickMentionScreenName {

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

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

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

2: required string handle(personalDataType = 'UserName')

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

struct ClientTweetReport {

/\*

\* Whether the "Report Tweet" flow was successfully completed.

\* `true` if the flow was completed successfully, `false` otherwise.

\*/

1: required bool isReportTweetDone

/\*

\* report-flow-id is included in Client Event when the "Report Tweet" flow was initiated

\* See go/report-flow-ids and

\* https://confluence.twitter.biz/pages/viewpage.action?spaceKey=HEALTH&title=Understanding+ReportDetails

\*/

2: optional string reportFlowId

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

enum TweetAuthorFollowClickSource {

UNKNOWN = 1

CARET\_MENU = 2

PROFILE\_IMAGE = 3

}

struct ClientTweetFollowAuthor {

/\*

\* Where did the user click the Follow button on the tweet - from the caret menu("CARET\_MENU")

\* or via hovering over the profile and clicking on Follow ("PROFILE\_IMAGE") - only applicable for web clients

\* "UNKNOWN" if the scribe do not match the expected namespace for the above

\*/

1: required TweetAuthorFollowClickSource followClickSource

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

enum TweetAuthorUnfollowClickSource {

UNKNOWN = 1

CARET\_MENU = 2

PROFILE\_IMAGE = 3

}

struct ClientTweetUnfollowAuthor {

/\*

\* Where did the user click the Unfollow button on the tweet - from the caret menu("CARET\_MENU")

\* or via hovering over the profile and clicking on Unfollow ("PROFILE\_IMAGE") - only applicable for web clients

\* "UNKNOWN" if the scribe do not match the expected namespace for the above

\*/

1: required TweetAuthorUnfollowClickSource unfollowClickSource

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

struct ServerTweetReport {

/\*

\* ReportDetails will be populated when the tweet report was scribed by spamacaw (server side)

\* Only for the action submit, all the fields under ReportDetails will be available.

\* This is because only after successful submission, we will know the report\_type and report\_flow\_name.

\* Reference: https://confluence.twitter.biz/pages/viewpage.action?spaceKey=HEALTH&title=Understanding+ReportDetails

\*/

1: optional string reportFlowId

2: optional report\_flow\_logs.ReportType reportType

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

/\*

\* This union will be updated when we have a particular

\* action that has attributes unique to that particular action

\* (e.g. linger impressions have start/end times) and not common

\* to other profile actions.

\*

\* Naming convention for ProfileActionInfo should be consistent with

\* ActionType. For example, `ClientProfileV2Impression` ActionType enum

\* should correspond to `ClientProfileV2Impression` ProfileActionInfo union arm.

\*/

union ProfileActionInfo {

// 56 matches enum index ServerProfileReport in ActionType

56: ServerProfileReport serverProfileReport

// 1600 matches enum index ClientProfileV2Impression in ActionType

1600: ClientProfileV2Impression clientProfileV2Impression

// 6001 matches enum index ServerUserUpdate in ActionType

6001: ServerUserUpdate serverUserUpdate

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

/\*

\* See go/behavioral-client-events for general behavioral client event (BCE) information

\* and https://docs.google.com/document/d/16CdSRpsmUUd17yoFH9min3nLBqDVawx4DaZoiqSfCHI/edit#heading=h.3tu05p92xgxc

\* for detailed information about BCE impression event.

\*

\* Unlike ClientTweetLingerImpression, there is no lower bound on the amount of time

\* necessary for the impress event to occur. There is also no visibility requirement for a impress

\* event to occur.

\*/

struct ClientProfileV2Impression {

/\* Milliseconds since epoch when the profile page became visible. \*/

1: required i64 impressStartTimestampMs(personalDataType = 'ImpressionMetadata')

/\* Milliseconds since epoch when the profile page became visible. \*/

2: required i64 impressEndTimestampMs(personalDataType = 'ImpressionMetadata')

/\*

\* The UI component that hosted this profile where the impress event happened.

\*

\* For example, sourceComponent = "profile" if the impress event happened on a profile page

\*/

3: required string sourceComponent(personalDataType = 'WebsitePage')

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

struct ServerProfileReport {

1: required social\_graph\_service\_write\_log.Action reportType(personalDataType = 'ReportType')

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

struct ServerUserUpdate {

1: required list<user\_service.UpdateDiffItem> updates

2: optional bool success (personalDataType = 'AuditMessage')

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