namespace java com.twitter.tsp.thriftjava

namespace py gen.twitter.tsp

#@namespace scala com.twitter.tsp.thriftscala

#@namespace strato com.twitter.tsp.strato

include "com/twitter/contentrecommender/common.thrift"

include "com/twitter/simclusters\_v2/identifier.thrift"

include "com/twitter/simclusters\_v2/online\_store.thrift"

include "topic\_listing.thrift"

enum TopicListingSetting {

All = 0 // All the existing Semantic Core Entity/Topics. ie., All topics on twitter, and may or may not have been launched yet.

Followable = 1 // All the topics which the user is allowed to follow. ie., topics that have shipped, and user may or may not be following it.

Following = 2 // Only topics the user is explicitly following

ImplicitFollow = 3 // The topics user has not followed but implicitly may follow. ie., Only topics that user has not followed.

} (hasPersonalData='false')

// used to tell Topic Social Proof endpoint which specific filtering can be bypassed

enum TopicSocialProofFilteringBypassMode {

NotInterested = 0

} (hasPersonalData='false')

struct TopicSocialProofRequest {

1: required i64 userId(personalDataType = "UserId")

2: required set<i64> tweetIds(personalDataType = 'TweetId')

3: required common.DisplayLocation displayLocation

4: required TopicListingSetting topicListingSetting

5: required topic\_listing.TopicListingViewerContext context

6: optional set<TopicSocialProofFilteringBypassMode> bypassModes

7: optional map<i64, set<MetricTag>> tags

}

struct TopicSocialProofOptions {

1: required i64 userId(personalDataType = "UserId")

2: required common.DisplayLocation displayLocation

3: required TopicListingSetting topicListingSetting

4: required topic\_listing.TopicListingViewerContext context

5: optional set<TopicSocialProofFilteringBypassMode> bypassModes

6: optional map<i64, set<MetricTag>> tags

}

struct TopicSocialProofResponse {

1: required map<i64, list<TopicWithScore>> socialProofs

}(hasPersonalData='false')

// Distinguishes between how a topic tweet is generated. Useful for metric tracking and debugging

enum TopicTweetType {

// CrOON candidates

UserInterestedIn = 1

Twistly = 2

// crTopic candidates

SkitConsumerEmbeddings = 100

SkitProducerEmbeddings = 101

SkitHighPrecision = 102

SkitInterestBrowser = 103

Certo = 104

}(persisted='true')

struct TopicWithScore {

1: required i64 topicId

2: required double score // score used to rank topics relative to one another

3: optional TopicTweetType algorithmType // how the topic is generated

4: optional TopicFollowType topicFollowType // Whether the topic is being explicitly or implicily followed

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

struct ScoreKey {

1: required identifier.EmbeddingType userEmbeddingType

2: required identifier.EmbeddingType topicEmbeddingType

3: required online\_store.ModelVersion modelVersion

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

struct UserTopicScore {

1: required map<ScoreKey, double> scores

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

enum TopicFollowType {

Following = 1

ImplicitFollow = 2

}(persisted='true')

// Provide the Tags which provides the Recommended Tweets Source Signal and other context.

// Warning: Please don't use this tag in any ML Features or business logic.

enum MetricTag {

// Source Signal Tags

TweetFavorite = 0

Retweet = 1

UserFollow = 101

PushOpenOrNtabClick = 201

HomeTweetClick = 301

HomeVideoView = 302

HomeSongbirdShowMore = 303

InterestsRankerRecentSearches = 401 // For Interests Candidate Expansion

UserInterestedIn = 501

MBCG = 503

// Other Metric Tags

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