namespace java com.twitter.cr\_mixer.thriftjava

#@namespace scala com.twitter.cr\_mixer.thriftscala

#@namespace strato com.twitter.cr\_mixer

include "ads.thrift"

include "candidate\_generation\_key.thrift"

include "cr\_mixer.thrift"

include "metric\_tags.thrift"

include "product.thrift"

include "related\_tweet.thrift"

include "source\_type.thrift"

include "uteg.thrift"

include "com/twitter/ml/api/data.thrift"

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

struct VITTweetCandidatesScribe {

1: required i64 uuid (personalDataType = 'UniversallyUniqueIdentifierUuid') # RequestUUID - unique scribe id for every request that comes in. Same request but different stages of scribe log (FetchCandidate, Filter, etc) share the same uuid

2: required i64 userId (personalDataType = 'UserId')

3: required list<VITTweetCandidateScribe> candidates

7: required product.Product product

8: required list<ImpressesedBucketInfo> impressedBuckets

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

struct VITTweetCandidateScribe {

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

2: required i64 authorId (personalDataType = 'UserId')

3: required double score

4: required list<metric\_tags.MetricTag> metricTags

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

struct GetTweetsRecommendationsScribe {

1: required i64 uuid (personalDataType = 'UniversallyUniqueIdentifierUuid') # RequestUUID - unique scribe id for every request that comes in. Same request but different stages of scribe log (FetchCandidate, Filter, etc) share the same uuid

2: required i64 userId (personalDataType = 'UserId')

3: required Result result

4: optional i64 traceId

5: optional PerformanceMetrics performanceMetrics

6: optional list<ImpressesedBucketInfo> impressedBuckets

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

struct SourceSignal {

# optional, since that the next step covers all info here

1: optional identifier.InternalId id

} (persisted='true')

struct PerformanceMetrics {

1: optional i64 latencyMs

} (persisted='true')

struct TweetCandidateWithMetadata {

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

2: optional candidate\_generation\_key.CandidateGenerationKey candidateGenerationKey

3: optional i64 authorId (personalDataType = 'UserId') # only for InterleaveResult for hydrating training data

4: optional double score # score with respect to candidateGenerationKey

5: optional data.DataRecord dataRecord # attach any features to this candidate

6: optional i32 numCandidateGenerationKeys # num CandidateGenerationKeys generating this tweetId

} (persisted='true')

struct FetchSignalSourcesResult {

1: optional set<SourceSignal> signals

} (persisted='true')

struct FetchCandidatesResult {

1: optional list<TweetCandidateWithMetadata> tweets

} (persisted='true')

struct PreRankFilterResult {

1: optional list<TweetCandidateWithMetadata> tweets

} (persisted='true')

struct InterleaveResult {

1: optional list<TweetCandidateWithMetadata> tweets

} (persisted='true')

struct RankResult {

1: optional list<TweetCandidateWithMetadata> tweets

} (persisted='true')

struct TopLevelApiResult {

1: required i64 timestamp (personalDataType = 'PrivateTimestamp')

2: required cr\_mixer.CrMixerTweetRequest request

3: required cr\_mixer.CrMixerTweetResponse response

} (persisted='true')

union Result {

1: FetchSignalSourcesResult fetchSignalSourcesResult

2: FetchCandidatesResult fetchCandidatesResult

3: PreRankFilterResult preRankFilterResult

4: InterleaveResult interleaveResult

5: RankResult rankResult

6: TopLevelApiResult topLevelApiResult

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

struct ImpressesedBucketInfo {

1: required i64 experimentId (personalDataType = 'ExperimentId')

2: required string bucketName

3: required i32 version

} (persisted='true')

############# RelatedTweets Scribe #############

struct GetRelatedTweetsScribe {

1: required i64 uuid (personalDataType = 'UniversallyUniqueIdentifierUuid') # RequestUUID - unique scribe id for every request that comes in. Same request but different stages of scribe log (FetchCandidate, Filter, etc) share the same uuid

2: required identifier.InternalId internalId

3: required RelatedTweetResult relatedTweetResult

4: optional i64 requesterId (personalDataType = 'UserId')

5: optional i64 guestId (personalDataType = 'GuestId')

6: optional i64 traceId

7: optional PerformanceMetrics performanceMetrics

8: optional list<ImpressesedBucketInfo> impressedBuckets

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

struct RelatedTweetTopLevelApiResult {

1: required i64 timestamp (personalDataType = 'PrivateTimestamp')

2: required related\_tweet.RelatedTweetRequest request

3: required related\_tweet.RelatedTweetResponse response

} (persisted='true')

union RelatedTweetResult {

1: RelatedTweetTopLevelApiResult relatedTweetTopLevelApiResult

2: FetchCandidatesResult fetchCandidatesResult

3: PreRankFilterResult preRankFilterResult # results after seqential filters

# if later we need rankResult, we can add it here

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

############# UtegTweets Scribe #############

struct GetUtegTweetsScribe {

1: required i64 uuid (personalDataType = 'UniversallyUniqueIdentifierUuid') # RequestUUID - unique scribe id for every request that comes in. Same request but different stages of scribe log (FetchCandidate, Filter, etc) share the same uuid

2: required i64 userId (personalDataType = 'UserId')

3: required UtegTweetResult utegTweetResult

4: optional i64 traceId

5: optional PerformanceMetrics performanceMetrics

6: optional list<ImpressesedBucketInfo> impressedBuckets

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

struct UtegTweetTopLevelApiResult {

1: required i64 timestamp (personalDataType = 'PrivateTimestamp')

2: required uteg.UtegTweetRequest request

3: required uteg.UtegTweetResponse response

} (persisted='true')

union UtegTweetResult {

1: UtegTweetTopLevelApiResult utegTweetTopLevelApiResult

2: FetchCandidatesResult fetchCandidatesResult

# if later we need rankResult, we can add it here

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

############# getAdsRecommendations() Scribe #############

struct GetAdsRecommendationsScribe {

1: required i64 uuid (personalDataType = 'UniversallyUniqueIdentifierUuid') # RequestUUID - unique scribe id for every request that comes in. Same request but different stages of scribe log (FetchCandidate, Filter, etc) share the same uuid

2: required i64 userId (personalDataType = 'UserId')

3: required AdsRecommendationsResult result

4: optional i64 traceId

5: optional PerformanceMetrics performanceMetrics

6: optional list<ImpressesedBucketInfo> impressedBuckets

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

struct AdsRecommendationTopLevelApiResult {

1: required i64 timestamp (personalDataType = 'PrivateTimestamp')

2: required ads.AdsRequest request

3: required ads.AdsResponse response

} (persisted='true')

union AdsRecommendationsResult{

1: AdsRecommendationTopLevelApiResult adsRecommendationTopLevelApiResult

2: FetchCandidatesResult fetchCandidatesResult

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