namespace java com.twitter.follow\_recommendations.thriftjava

#@namespace scala com.twitter.follow\_recommendations.thriftscala

#@namespace strato com.twitter.follow\_recommendations

include "assembler.thrift"

include "client\_context.thrift"

include "debug.thrift"

include "display\_context.thrift"

include "display\_location.thrift"

include "recommendations.thrift"

include "recently\_engaged\_user\_id.thrift"

include "finatra-thrift/finatra\_thrift\_exceptions.thrift"

include "com/twitter/product\_mixer/core/pipeline\_execution\_result.thrift"

struct RecommendationRequest {

1: required client\_context.ClientContext clientContext

2: required display\_location.DisplayLocation displayLocation

3: optional display\_context.DisplayContext displayContext

// Max results to return

4: optional i32 maxResults

// Cursor to continue returning results if any

5: optional string cursor

// IDs of Content to exclude from recommendations

6: optional list<i64> excludedIds(personalDataType='UserId')

// Whether to also get promoted content

7: optional bool fetchPromotedContent

8: optional debug.DebugParams debugParams

9: optional string userLocationState(personalDataType='InferredLocation')

}(hasPersonalData='true')

struct RecommendationResponse {

1: required list<recommendations.Recommendation> recommendations

}(hasPersonalData='true')

// for scoring a list of candidates, while logging hydrated features

struct ScoringUserRequest {

1: required client\_context.ClientContext clientContext

2: required display\_location.DisplayLocation displayLocation

3: required list<recommendations.UserRecommendation> candidates

4: optional debug.DebugParams debugParams

}(hasPersonalData='true')

struct ScoringUserResponse {

1: required list<recommendations.UserRecommendation> candidates // empty for now

}(hasPersonalData='true')

// for getting the list of candidates generated by a single candidate source

struct DebugCandidateSourceRequest {

1: required client\_context.ClientContext clientContext

2: required debug.DebugCandidateSourceIdentifier candidateSource

3: optional list<i64> uttInterestIds

4: optional debug.DebugParams debugParams

5: optional list<i64> recentlyFollowedUserIds

6: optional list<recently\_engaged\_user\_id.RecentlyEngagedUserId> recentlyEngagedUserIds

7: optional list<i64> byfSeedUserIds

8: optional list<i64> similarToUserIds

9: required bool applySgsPredicate

10: optional i32 maxResults

}(hasPersonalData='true')

service FollowRecommendationsThriftService {

RecommendationResponse getRecommendations(1: RecommendationRequest request) throws (

1: finatra\_thrift\_exceptions.ServerError serverError,

2: finatra\_thrift\_exceptions.UnknownClientIdError unknownClientIdError,

3: finatra\_thrift\_exceptions.NoClientIdError noClientIdError

)

RecommendationDisplayResponse getRecommendationDisplayResponse(1: RecommendationRequest request) throws (

1: finatra\_thrift\_exceptions.ServerError serverError,

2: finatra\_thrift\_exceptions.UnknownClientIdError unknownClientIdError,

3: finatra\_thrift\_exceptions.NoClientIdError noClientIdError

)

// temporary endpoint for feature hydration and logging for data collection.

ScoringUserResponse scoreUserCandidates(1: ScoringUserRequest request) throws (

1: finatra\_thrift\_exceptions.ServerError serverError,

2: finatra\_thrift\_exceptions.UnknownClientIdError unknownClientIdError,

3: finatra\_thrift\_exceptions.NoClientIdError noClientIdError

)

// Debug endpoint for getting recommendations of a single candidate source. We can remove this endpoint when ProMix provide this functionality and we integrate with it.

RecommendationResponse debugCandidateSource(1: DebugCandidateSourceRequest request) throws (

1: finatra\_thrift\_exceptions.ServerError serverError,

2: finatra\_thrift\_exceptions.UnknownClientIdError unknownClientIdError,

3: finatra\_thrift\_exceptions.NoClientIdError noClientIdError

)

// Get the full execution log for a pipeline (used by our debugging tools)

pipeline\_execution\_result.PipelineExecutionResult executePipeline(1: RecommendationRequest request) throws (

1: finatra\_thrift\_exceptions.ServerError serverError,

2: finatra\_thrift\_exceptions.UnknownClientIdError unknownClientIdError,

3: finatra\_thrift\_exceptions.NoClientIdError noClientIdError

)

}

struct RecommendationDisplayResponse {

1: required list<recommendations.HydratedRecommendation> hydratedRecommendation

2: optional assembler.Header header

3: optional assembler.Footer footer

4: optional assembler.WTFPresentation wtfPresentation

}(hasPersonalData='true')