package com.twitter.tweetypie.federated

package columns

import com.twitter.bouncer.thriftscala.Bounce

import com.twitter.finagle.http.Status

import com.twitter.finatra.api11

import com.twitter.finatra.api11.ApiError

import com.twitter.strato.response.Err

object ApiErrors {

// Errs ported from StatusesRetweetController

val GenericAccessDeniedErr = toErr(ApiError.GenericAccessDenied)

val AlreadyRetweetedErr = toErr(ApiError.AlreadyRetweeted)

val DuplicateStatusErr = toErr(ApiError.DuplicateStatusError)

val InvalidRetweetForStatusErr = toErr(ApiError.InvalidRetweetForStatus)

val StatusNotFoundErr = toErr(ApiError.StatusNotFound)

val BlockedUserErr =

toErr(ApiError.BlockedUserError, "retweeting this user's tweets at their request")

val ClientNotPrivilegedErr = toErr(ApiError.ClientNotPrivileged)

val UserDeniedRetweetErr = toErr(ApiError.CurrentUserSuspended)

// Errs ported from StatusesUpdateController

val RateLimitExceededErr = toErr(ApiError.OverStatusUpdateLimit, "User")

val TweetUrlSpamErr = toErr(ApiError.TieredActionTweetUrlSpam)

val TweetSpammerErr = toErr(ApiError.TieredActionTweetSpammer)

val CaptchaChallengeErr = toErr(ApiError.TieredActionChallengeCaptcha)

val SafetyRateLimitExceededErr = toErr(ApiError.UserActionRateLimitExceeded, "User")

val TweetCannotBeBlankErr = // was MissingRequiredParameterException

toErr(ApiError.ForbiddenMissingParameter, "tweet\_text or media")

val TweetTextTooLongErr = toErr(ApiError.StatusTooLongError)

val MalwareTweetErr = toErr(ApiError.StatusMalwareError)

val DuplicateTweetErr = toErr(ApiError.DuplicateStatusError)

val CurrentUserSuspendedErr = toErr(ApiError.CurrentUserSuspended)

val MentionLimitExceededErr = toErr(ApiError.MentionLimitInTweetExceeded)

val UrlLimitExceededErr = toErr(ApiError.UrlLimitInTweetExceeded)

val HashtagLimitExceededErr = toErr(ApiError.HashtagLimitInTweetExceeded)

val CashtagLimitExceededErr = toErr(ApiError.CashtagLimitInTweetExceeded)

val HashtagLengthLimitExceededErr = toErr(ApiError.HashtagLengthLimitInTweetExceeded)

val TooManyAttachmentTypesErr = toErr(ApiError.AttachmentTypesLimitInTweetExceeded)

val InvalidAttachmentUrlErr = toErr(ApiError.InvalidParameter("attachment\_url"))

val InReplyToTweetNotFoundErr = toErr(ApiError.InReplyToTweetNotFound)

val InvalidAdditionalFieldErr = toErr(ApiError.GenericBadRequest)

def invalidAdditionalFieldWithReasonErr(failureReason: String) =

toErr(ApiError.GenericBadRequest.copy(message = failureReason))

val InvalidUrlErr = toErr(ApiError.InvalidUrl)

val InvalidCoordinatesErr = toErr(ApiError.InvalidCoordinates)

val InvalidGeoSearchRequestIdErr =

toErr(ApiError.InvalidParameter("geo\_search\_request\_id"))

val ConversationControlNotAuthorizedErr = toErr(ApiError.ConversationControlNotAuthorized)

val ConversationControlInvalidErr = toErr(ApiError.ConversationControlInvalid)

val ConversationControlReplyRestricted = toErr(ApiError.ConversationControlReplyRestricted)

// Errors ported from StatusesDestroyController

val DeletePermissionErr = toErr(ApiError.StatusActionPermissionError("delete"))

// See StatusesUpdateController#GenericErrorException

val GenericTweetCreateErr = toErr(ApiError.UnknownInterpreterError, "Tweet creation failed")

val InvalidBatchModeParameterErr = toErr(ApiError.InvalidParameter("batch\_mode"))

val CannotConvoControlAndCommunitiesErr =

toErr(ApiError.CommunityInvalidParams, "conversation\_control")

val TooManyCommunitiesErr = toErr(ApiError.CommunityInvalidParams, "communities")

val CommunityReplyTweetNotAllowedErr = toErr(ApiError.CommunityReplyTweetNotAllowed)

val ConversationControlNotSupportedErr = toErr(ApiError.ConversationControlNotSupported)

val CommunityUserNotAuthorizedErr = toErr(ApiError.CommunityUserNotAuthorized)

val CommunityNotFoundErr = toErr(ApiError.CommunityNotFound)

val CommunityProtectedUserCannotTweetErr = toErr(ApiError.CommunityProtectedUserCannotTweet)

val SuperFollowCreateNotAuthorizedErr = toErr(ApiError.SuperFollowsCreateNotAuthorized)

val SuperFollowInvalidParamsErr = toErr(ApiError.SuperFollowsInvalidParams)

val ExclusiveTweetEngagementNotAllowedErr = toErr(ApiError.ExclusiveTweetEngagementNotAllowed)

val SafetyLevelMissingErr = toErr(ApiError.MissingParameter("safety\_level"))

def accessDeniedByBouncerErr(bounce: Bounce) =

toErr(ApiError.AccessDeniedByBouncer, bounce.errorMessage.getOrElse(Seq.empty))

def tweetEngagementLimitedErr(failureReason: String) =

toErr(ApiError.TweetEngagementsLimited(failureReason))

def invalidMediaErr(failureReason: Option[String]) =

toErr(ApiError.invalidMediaId(failureReason))

val TrustedFriendsInvalidParamsErr = toErr(ApiError.TrustedFriendsInvalidParams)

val TrustedFriendsRetweetNotAllowedErr = toErr(ApiError.TrustedFriendsRetweetNotAllowed)

val TrustedFriendsEngagementNotAllowedErr = toErr(ApiError.TrustedFriendsEngagementNotAllowed)

val TrustedFriendsCreateNotAllowedErr = toErr(ApiError.TrustedFriendsCreateNotAllowed)

val TrustedFriendsQuoteTweetNotAllowedErr = toErr(ApiError.TrustedFriendsQuoteTweetNotAllowed)

val StaleTweetEngagementNotAllowedErr = toErr(ApiError.StaleTweetEngagementNotAllowed)

val StaleTweetQuoteTweetNotAllowedErr = toErr(ApiError.StaleTweetQuoteTweetNotAllowed)

val StaleTweetRetweetNotAllowedErr = toErr(ApiError.StaleTweetRetweetNotAllowed)

val CollabTweetInvalidParamsErr = toErr(ApiError.CollabTweetInvalidParams)

val FieldEditNotAllowedErr = toErr(ApiError.FieldEditNotAllowed)

val NotEligibleForEditErr = toErr(ApiError.NotEligibleForEdit)

def toErr(apiError: api11.ApiError, args: Any\*): Err = {

val errCode = apiError.status match {

case Status.Forbidden => Err.Authorization

case Status.Unauthorized => Err.Authentication

case Status.NotFound => Err.BadRequest

case Status.BadRequest => Err.BadRequest

case \_ => Err.BadRequest

}

val errMessage = s"${apiError.message.format(args.mkString(","))} (${apiError.code})"

val errContext = Some(Err.Context.Api11Error(apiError.code))

Err(errCode, errMessage, errContext)

}

}