package com.twitter.visibility.rules

import com.twitter.visibility.common.actions.LimitedEngagementReason

import com.twitter.visibility.configapi.params.FSRuleParams.AdAvoidanceHighToxicityModelScoreThresholdParam

import com.twitter.visibility.configapi.params.FSRuleParams.AdAvoidanceReportedTweetModelScoreThresholdParam

import com.twitter.visibility.configapi.params.FSRuleParams.CommunityTweetCommunityUnavailableLimitedActionsRulesEnabledParam

import com.twitter.visibility.configapi.params.FSRuleParams.CommunityTweetDropProtectedRuleEnabledParam

import com.twitter.visibility.configapi.params.FSRuleParams.CommunityTweetDropRuleEnabledParam

import com.twitter.visibility.configapi.params.FSRuleParams.CommunityTweetLimitedActionsRulesEnabledParam

import com.twitter.visibility.configapi.params.FSRuleParams.CommunityTweetMemberRemovedLimitedActionsRulesEnabledParam

import com.twitter.visibility.configapi.params.FSRuleParams.CommunityTweetNonMemberLimitedActionsRuleEnabledParam

import com.twitter.visibility.configapi.params.FSRuleParams.StaleTweetLimitedActionsRulesEnabledParam

import com.twitter.visibility.configapi.params.FSRuleParams.TrustedFriendsTweetLimitedEngagementsRuleEnabledParam

import com.twitter.visibility.configapi.params.RuleParam

import com.twitter.visibility.configapi.params.RuleParams

import com.twitter.visibility.configapi.params.RuleParams.\_

import com.twitter.visibility.features.{TweetDeleteReason => FeatureTweetDeleteReason}

import com.twitter.visibility.models.TweetDeleteReason

import com.twitter.visibility.models.TweetSafetyLabelType

import com.twitter.visibility.rules.Condition.ViewerIsExclusiveTweetAuthor

import com.twitter.visibility.rules.Condition.\_

import com.twitter.visibility.rules.Reason.CommunityTweetAuthorRemoved

import com.twitter.visibility.rules.Reason.CommunityTweetHidden

import com.twitter.visibility.rules.Reason.Nsfw

import com.twitter.visibility.rules.Reason.StaleTweet

import com.twitter.visibility.rules.Reason.Unspecified

import com.twitter.visibility.rules.RuleActionSourceBuilder.TweetSafetyLabelSourceBuilder

abstract class TweetHasLabelRule(action: Action, tweetSafetyLabelType: TweetSafetyLabelType)

extends RuleWithConstantAction(action, TweetHasLabel(tweetSafetyLabelType)) {

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(tweetSafetyLabelType))

}

abstract class ConditionWithTweetLabelRule(

action: Action,

condition: Condition,

tweetSafetyLabelType: TweetSafetyLabelType)

extends RuleWithConstantAction(action, And(TweetHasLabel(tweetSafetyLabelType), condition)) {

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(tweetSafetyLabelType))

}

abstract class NonAuthorWithTweetLabelRule(

action: Action,

tweetSafetyLabelType: TweetSafetyLabelType)

extends ConditionWithTweetLabelRule(action, NonAuthorViewer, tweetSafetyLabelType) {

override def actionSourceBuilder: Option[RuleActionSourceBuilder] = Some(

TweetSafetyLabelSourceBuilder(tweetSafetyLabelType))

}

abstract class NonFollowerWithTweetLabelRule(

action: Action,

tweetSafetyLabelType: TweetSafetyLabelType)

extends ConditionWithTweetLabelRule(

action,

LoggedOutOrViewerNotFollowingAuthor,

tweetSafetyLabelType

)

abstract class NonAuthorAndNonFollowerWithTweetLabelRule(

action: Action,

tweetSafetyLabelType: TweetSafetyLabelType)

extends ConditionWithTweetLabelRule(

action,

And(NonAuthorViewer, LoggedOutOrViewerNotFollowingAuthor),

tweetSafetyLabelType

)

abstract class NonFollowerWithUqfTweetLabelRule(

action: Action,

tweetSafetyLabelType: TweetSafetyLabelType)

extends ConditionWithTweetLabelRule(

action,

Or(

LoggedOutViewer,

And(

NonAuthorViewer,

Not(ViewerDoesFollowAuthor),

ViewerHasUqfEnabled

)

),

tweetSafetyLabelType

)

abstract class ViewerWithUqfTweetLabelRule(action: Action, labelValue: TweetSafetyLabelType)

extends ConditionWithTweetLabelRule(action, ViewerHasUqfEnabled, labelValue)

case object ConversationControlRules {

abstract class ConversationControlBaseRule(condition: Condition)

extends RuleWithConstantAction(

LimitedEngagements(LimitedEngagementReason.ConversationControl),

condition) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(TweetConversationControlEnabledParam)

}

object LimitRepliesCommunityConversationRule

extends ConversationControlBaseRule(

And(

TweetIsCommunityConversation,

Not(

Or(

LoggedOutViewer,

Retweet,

ViewerIsTweetConversationRootAuthor,

ViewerIsInvitedToTweetConversation,

ConversationRootAuthorDoesFollowViewer

))

)

)

object LimitRepliesFollowersConversationRule

extends ConversationControlBaseRule(

And(

TweetIsFollowersConversation,

Not(

Or(

LoggedOutViewer,

Retweet,

ViewerIsTweetConversationRootAuthor,

ViewerIsInvitedToTweetConversation,

ViewerDoesFollowConversationRootAuthor

))

)

)

object LimitRepliesByInvitationConversationRule

extends ConversationControlBaseRule(

And(

TweetIsByInvitationConversation,

Not(

Or(

LoggedOutViewer,

Retweet,

ViewerIsTweetConversationRootAuthor,

ViewerIsInvitedToTweetConversation

))

)

)

}

abstract class NonAuthorViewerOptInFilteringWithTweetLabelRule(

action: Action,

tweetSafetyLabelType: TweetSafetyLabelType)

extends ConditionWithTweetLabelRule(

action,

And(NonAuthorViewer, LoggedOutOrViewerOptInFiltering),

tweetSafetyLabelType)

abstract class NonFollowerViewerOptInFilteringWithTweetLabelRule(

action: Action,

tweetSafetyLabelType: TweetSafetyLabelType)

extends ConditionWithTweetLabelRule(

action,

And(LoggedOutOrViewerNotFollowingAuthor, LoggedOutOrViewerOptInFiltering),

tweetSafetyLabelType

)

object TweetNsfwUserDropRule extends RuleWithConstantAction(Drop(Nsfw), TweetHasNsfwUserAuthor)

object TweetNsfwAdminDropRule extends RuleWithConstantAction(Drop(Nsfw), TweetHasNsfwAdminAuthor)

object NullcastedTweetRule

extends RuleWithConstantAction(

Drop(Unspecified),

And(Nullcast, Not(Retweet), Not(IsQuotedInnerTweet), Not(TweetIsCommunityTweet)))

object MutedRetweetsRule

extends RuleWithConstantAction(Drop(Unspecified), And(Retweet, ViewerMutesRetweetsFromAuthor))

abstract class FilterCommunityTweetsRule(override val action: Action)

extends RuleWithConstantAction(action, TweetIsCommunityTweet) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(CommunityTweetDropRuleEnabledParam)

}

object DropCommunityTweetsRule extends FilterCommunityTweetsRule(Drop(CommunityTweetHidden))

object TombstoneCommunityTweetsRule

extends FilterCommunityTweetsRule(Tombstone(Epitaph.Unavailable))

abstract class FilterCommunityTweetCommunityNotVisibleRule(override val action: Action)

extends RuleWithConstantAction(

action,

And(

NonAuthorViewer,

TweetIsCommunityTweet,

Not(CommunityTweetCommunityVisible),

)) {

override def enabled: Seq[RuleParam[Boolean]] =

Seq(DropCommunityTweetWithUndefinedCommunityRuleEnabledParam)

}

object DropCommunityTweetCommunityNotVisibleRule

extends FilterCommunityTweetCommunityNotVisibleRule(Drop(CommunityTweetHidden))

object TombstoneCommunityTweetCommunityNotVisibleRule

extends FilterCommunityTweetCommunityNotVisibleRule(Tombstone(Epitaph.Unavailable))

abstract class FilterAllCommunityTweetsRule(override val action: Action)

extends RuleWithConstantAction(action, TweetIsCommunityTweet) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(CommunityTweetsEnabledParam)

}

object DropAllCommunityTweetsRule extends FilterAllCommunityTweetsRule(Drop(Unspecified))

object TombstoneAllCommunityTweetsRule

extends FilterAllCommunityTweetsRule(Tombstone(Epitaph.Unavailable))

object DropOuterCommunityTweetsRule

extends RuleWithConstantAction(

Drop(Unspecified),

And(TweetIsCommunityTweet, Not(IsQuotedInnerTweet)))

object DropAllHiddenCommunityTweetsRule

extends RuleWithConstantAction(

Drop(Unspecified),

And(TweetIsCommunityTweet, CommunityTweetIsHidden))

abstract class FilterHiddenCommunityTweetsRule(override val action: Action)

extends RuleWithConstantAction(

action,

And(

NonAuthorViewer,

TweetIsCommunityTweet,

CommunityTweetIsHidden,

Not(ViewerIsCommunityModerator)

))

object DropHiddenCommunityTweetsRule

extends FilterHiddenCommunityTweetsRule(Drop(CommunityTweetHidden))

object TombstoneHiddenCommunityTweetsRule

extends FilterHiddenCommunityTweetsRule(Tombstone(Epitaph.CommunityTweetHidden))

object DropAllAuthorRemovedCommunityTweetsRule

extends RuleWithConstantAction(

Drop(Unspecified),

And(TweetIsCommunityTweet, CommunityTweetAuthorIsRemoved))

abstract class FilterAuthorRemovedCommunityTweetsRule(override val action: Action)

extends RuleWithConstantAction(

action,

And(

NonAuthorViewer,

TweetIsCommunityTweet,

CommunityTweetAuthorIsRemoved,

Not(ViewerIsCommunityModerator)

))

object DropAuthorRemovedCommunityTweetsRule

extends FilterAuthorRemovedCommunityTweetsRule(Drop(CommunityTweetAuthorRemoved))

object TombstoneAuthorRemovedCommunityTweetsRule

extends FilterAuthorRemovedCommunityTweetsRule(Tombstone(Epitaph.Unavailable))

abstract class FilterProtectedCommunityTweetsRule(override val action: Action)

extends RuleWithConstantAction(

action,

And(TweetIsCommunityTweet, ProtectedAuthor, NonAuthorViewer)) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(CommunityTweetDropProtectedRuleEnabledParam)

}

object DropProtectedCommunityTweetsRule

extends FilterProtectedCommunityTweetsRule(Drop(CommunityTweetHidden))

object TombstoneProtectedCommunityTweetsRule

extends FilterProtectedCommunityTweetsRule(Tombstone(Epitaph.Unavailable))

abstract class CommunityTweetCommunityUnavailableLimitedActionsRule(

reason: LimitedEngagementReason,

condition: CommunityTweetCommunityUnavailable,

) extends RuleWithConstantAction(

LimitedEngagements(reason),

And(

Not(NonAuthorViewer),

TweetIsCommunityTweet,

condition,

)

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(

CommunityTweetCommunityUnavailableLimitedActionsRulesEnabledParam)

}

object CommunityTweetCommunityNotFoundLimitedActionsRule

extends CommunityTweetCommunityUnavailableLimitedActionsRule(

LimitedEngagementReason.CommunityTweetCommunityNotFound,

CommunityTweetCommunityNotFound,

)

object CommunityTweetCommunityDeletedLimitedActionsRule

extends CommunityTweetCommunityUnavailableLimitedActionsRule(

LimitedEngagementReason.CommunityTweetCommunityDeleted,

CommunityTweetCommunityDeleted,

)

object CommunityTweetCommunitySuspendedLimitedActionsRule

extends CommunityTweetCommunityUnavailableLimitedActionsRule(

LimitedEngagementReason.CommunityTweetCommunitySuspended,

CommunityTweetCommunitySuspended,

)

abstract class CommunityTweetModeratedLimitedActionsRule(

reason: LimitedEngagementReason,

condition: CommunityTweetIsModerated,

enabledParam: RuleParam[Boolean],

) extends RuleWithConstantAction(

LimitedEngagements(reason),

And(

TweetIsCommunityTweet,

condition,

Or(

Not(NonAuthorViewer),

ViewerIsCommunityModerator,

)

)) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(enabledParam)

}

object CommunityTweetMemberRemovedLimitedActionsRule

extends CommunityTweetModeratedLimitedActionsRule(

LimitedEngagementReason.CommunityTweetMemberRemoved,

CommunityTweetAuthorIsRemoved,

CommunityTweetMemberRemovedLimitedActionsRulesEnabledParam,

)

object CommunityTweetHiddenLimitedActionsRule

extends CommunityTweetModeratedLimitedActionsRule(

LimitedEngagementReason.CommunityTweetHidden,

CommunityTweetIsHidden,

CommunityTweetLimitedActionsRulesEnabledParam,

)

abstract class CommunityTweetLimitedActionsRule(

reason: LimitedEngagementReason,

condition: Condition,

) extends RuleWithConstantAction(

LimitedEngagements(reason),

And(

TweetIsCommunityTweet,

condition

)) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(CommunityTweetLimitedActionsRulesEnabledParam)

}

object CommunityTweetMemberLimitedActionsRule

extends CommunityTweetLimitedActionsRule(

LimitedEngagementReason.CommunityTweetMember,

ViewerIsCommunityMember,

)

object CommunityTweetNonMemberLimitedActionsRule

extends CommunityTweetLimitedActionsRule(

LimitedEngagementReason.CommunityTweetNonMember,

Not(ViewerIsCommunityMember),

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(

CommunityTweetNonMemberLimitedActionsRuleEnabledParam)

}

object ReportedTweetInterstitialRule

extends RuleWithConstantAction(

Interstitial(Reason.ViewerReportedTweet),

And(

NonAuthorViewer,

Not(Retweet),

ViewerReportsTweet

)) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableReportedTweetInterstitialRule)

}

object ReportedTweetInterstitialSearchRule

extends RuleWithConstantAction(

Interstitial(Reason.ViewerReportedTweet),

And(

NonAuthorViewer,

Not(Retweet),

ViewerReportsTweet

)) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableReportedTweetInterstitialSearchRule)

}

abstract class FilterExclusiveTweetContentRule(

action: Action,

additionalCondition: Condition = Condition.True)

extends RuleWithConstantAction(

action,

And(

additionalCondition,

TweetIsExclusiveContent,

Or(

LoggedOutViewer,

Not(

Or(

ViewerIsExclusiveTweetAuthor,

ViewerSuperFollowsExclusiveTweetAuthor,

And(

Not(NonAuthorViewer),

Not(Retweet)

)

)

),

),

)

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableDropExclusiveTweetContentRule)

override def enableFailClosed: Seq[RuleParam[Boolean]] = Seq(

EnableDropExclusiveTweetContentRuleFailClosed)

}

object DropExclusiveTweetContentRule

extends FilterExclusiveTweetContentRule(Drop(Reason.ExclusiveTweet))

object TombstoneExclusiveTweetContentRule

extends FilterExclusiveTweetContentRule(Tombstone(Epitaph.SuperFollowsContent))

object TombstoneExclusiveQuotedTweetContentRule

extends FilterExclusiveTweetContentRule(

Tombstone(Epitaph.SuperFollowsContent),

IsQuotedInnerTweet

)

object DropAllExclusiveTweetsRule

extends RuleWithConstantAction(

Drop(Reason.ExclusiveTweet),

TweetIsExclusiveContent

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableDropAllExclusiveTweetsRuleParam)

override def enableFailClosed: Seq[RuleParam[Boolean]] = Seq(

EnableDropAllExclusiveTweetsRuleFailClosedParam)

}

object DropTweetsWithGeoRestrictedMediaRule

extends RuleWithConstantAction(Drop(Unspecified), MediaRestrictedInViewerCountry) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(

EnableDropTweetsWithGeoRestrictedMediaRuleParam)

}

object TrustedFriendsTweetLimitedEngagementsRule

extends RuleWithConstantAction(

LimitedEngagements(LimitedEngagementReason.TrustedFriendsTweet),

TweetIsTrustedFriendsContent

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(

TrustedFriendsTweetLimitedEngagementsRuleEnabledParam

)

}

object DropAllTrustedFriendsTweetsRule

extends RuleWithConstantAction(

Drop(Reason.TrustedFriendsTweet),

TweetIsTrustedFriendsContent

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableDropAllTrustedFriendsTweetsRuleParam)

override def enableFailClosed: Seq[RuleParam[Boolean]] = Seq(RuleParams.True)

}

object DropAllCollabInvitationTweetsRule

extends RuleWithConstantAction(

Drop(Unspecified),

TweetIsCollabInvitationContent

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableDropAllCollabInvitationTweetsRuleParam)

override def enableFailClosed: Seq[RuleParam[Boolean]] = Seq(RuleParams.True)

}

abstract class FilterTrustedFriendsTweetContentRule(action: Action)

extends OnlyWhenNotAuthorViewerRule(

action,

And(

TweetIsTrustedFriendsContent,

Not(

Or(

ViewerIsTrustedFriendsTweetAuthor,

ViewerIsTrustedFriend

)

)

)

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableDropTrustedFriendsTweetContentRuleParam)

override def enableFailClosed: Seq[RuleParam[Boolean]] = Seq(RuleParams.True)

}

object DropTrustedFriendsTweetContentRule

extends FilterTrustedFriendsTweetContentRule(Drop(Reason.TrustedFriendsTweet))

object TombstoneTrustedFriendsTweetContentRule

extends FilterTrustedFriendsTweetContentRule(Tombstone(Epitaph.Unavailable))

object TweetNsfwUserAdminAvoidRule

extends RuleWithConstantAction(

Avoid(),

Or(

TweetHasNsfwUserAuthor,

TweetHasNsfwAdminAuthor,

NsfwUserAuthor,

NsfwAdminAuthor

)

)

object AvoidHighToxicityModelScoreRule

extends RuleWithConstantAction(

Avoid(),

TweetHasLabelWithScoreAboveThresholdWithParam(

TweetSafetyLabelType.HighToxicityScore,

AdAvoidanceHighToxicityModelScoreThresholdParam)

)

object AvoidReportedTweetModelScoreRule

extends RuleWithConstantAction(

Avoid(),

TweetHasLabelWithScoreAboveThresholdWithParam(

TweetSafetyLabelType.HighPReportedTweetScore,

AdAvoidanceReportedTweetModelScoreThresholdParam)

)

object TombstoneDeletedOuterTweetRule

extends RuleWithConstantAction(

Tombstone(Epitaph.Deleted),

And(

Equals(FeatureTweetDeleteReason, TweetDeleteReason.Deleted),

Not(IsQuotedInnerTweet)

)

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableDeleteStateTweetRulesParam)

}

object TombstoneDeletedTweetRule

extends RuleWithConstantAction(

Tombstone(Epitaph.Deleted),

And(

Equals(FeatureTweetDeleteReason, TweetDeleteReason.Deleted),

)

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableDeleteStateTweetRulesParam)

}

object TombstoneDeletedQuotedTweetRule

extends RuleWithConstantAction(

Tombstone(Epitaph.Deleted),

And(

Equals(FeatureTweetDeleteReason, TweetDeleteReason.Deleted),

IsQuotedInnerTweet

)

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableDeleteStateTweetRulesParam)

}

object TombstoneBounceDeletedTweetRule

extends RuleWithConstantAction(

Tombstone(Epitaph.BounceDeleted),

Equals(FeatureTweetDeleteReason, TweetDeleteReason.BounceDeleted),

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableDeleteStateTweetRulesParam)

}

object TombstoneBounceDeletedOuterTweetRule

extends RuleWithConstantAction(

Tombstone(Epitaph.BounceDeleted),

And(

Equals(FeatureTweetDeleteReason, TweetDeleteReason.BounceDeleted),

Not(IsQuotedInnerTweet)

)

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableDeleteStateTweetRulesParam)

}

object TombstoneBounceDeletedQuotedTweetRule

extends RuleWithConstantAction(

Tombstone(Epitaph.BounceDeleted),

And(

Equals(FeatureTweetDeleteReason, TweetDeleteReason.BounceDeleted),

IsQuotedInnerTweet

)

) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableDeleteStateTweetRulesParam)

}

object DropStaleTweetsRule

extends RuleWithConstantAction(

Drop(StaleTweet),

And(TweetIsStaleTweet, Not(IsQuotedInnerTweet), Not(Retweet), Not(IsSourceTweet))) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(EnableStaleTweetDropRuleParam)

override def enableFailClosed: Seq[RuleParam[Boolean]] = Seq(

EnableStaleTweetDropRuleFailClosedParam)

}

object StaleTweetLimitedActionsRule

extends RuleWithConstantAction(

LimitedEngagements(LimitedEngagementReason.StaleTweet),

TweetIsStaleTweet) {

override def enabled: Seq[RuleParam[Boolean]] = Seq(StaleTweetLimitedActionsRulesEnabledParam)

}