package com.twitter.visibility.rules

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

import com.twitter.visibility.rules.Condition.DeactivatedAuthor

import com.twitter.visibility.rules.Condition.ErasedAuthor

import com.twitter.visibility.rules.Condition.SuspendedAuthor

import com.twitter.visibility.rules.Condition.DmEventInOneToOneConversationWithUnavailableUser

import com.twitter.visibility.rules.Condition.DmEventIsBeforeLastClearedEvent

import com.twitter.visibility.rules.Condition.DmEventIsBeforeJoinConversationEvent

import com.twitter.visibility.rules.Condition.DmEventIsDeleted

import com.twitter.visibility.rules.Condition.DmEventIsHidden

import com.twitter.visibility.rules.Condition.LastMessageReadUpdateDmEvent

import com.twitter.visibility.rules.Condition.MessageCreateDmEvent

import com.twitter.visibility.rules.Condition.PerspectivalJoinConversationDmEvent

import com.twitter.visibility.rules.Condition.ViewerIsDmEventInitiatingUser

import com.twitter.visibility.rules.Condition.ViewerIsDmConversationParticipant

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

import com.twitter.visibility.rules.Condition.And

import com.twitter.visibility.rules.Condition.CsFeedbackDismissedDmEvent

import com.twitter.visibility.rules.Condition.CsFeedbackSubmittedDmEvent

import com.twitter.visibility.rules.Condition.JoinConversationDmEvent

import com.twitter.visibility.rules.Condition.Not

import com.twitter.visibility.rules.Condition.Or

import com.twitter.visibility.rules.Condition.TrustConversationDmEvent

import com.twitter.visibility.rules.Condition.WelcomeMessageCreateDmEvent

import com.twitter.visibility.rules.Condition.DmEventInOneToOneConversation

import com.twitter.visibility.rules.Condition.ConversationCreateDmEvent

object DmEventRules {

object MessageCreateEventWithUnavailableSenderDropRule

extends RuleWithConstantAction(

Drop(Unspecified),

Or(SuspendedAuthor, DeactivatedAuthor, ErasedAuthor)) {

override def enableFailClosed = Seq(RuleParams.True)

}

object WelcomeMessageCreateEventOnlyVisibleToRecipientDropRule

extends RuleWithConstantAction(

Drop(Unspecified),

And(ViewerIsDmEventInitiatingUser, WelcomeMessageCreateDmEvent)) {

override def enableFailClosed = Seq(RuleParams.True)

}

object InaccessibleDmEventDropRule

extends RuleWithConstantAction(

Drop(Unspecified),

Or(

Not(ViewerIsDmConversationParticipant),

DmEventIsBeforeLastClearedEvent,

DmEventIsBeforeJoinConversationEvent)) {

override def enableFailClosed = Seq(RuleParams.True)

}

object HiddenAndDeletedDmEventDropRule

extends RuleWithConstantAction(Drop(Unspecified), Or(DmEventIsDeleted, DmEventIsHidden)) {

override def enableFailClosed = Seq(RuleParams.True)

}

object NonPerspectivalDmEventDropRule

extends RuleWithConstantAction(

Drop(Unspecified),

Or(

And(Not(PerspectivalJoinConversationDmEvent), JoinConversationDmEvent),

And(

Not(ViewerIsDmEventInitiatingUser),

Or(TrustConversationDmEvent, CsFeedbackSubmittedDmEvent, CsFeedbackDismissedDmEvent))

)

) {

override def enableFailClosed = Seq(RuleParams.True)

}

object DmEventInOneToOneConversationWithUnavailableUserDropRule

extends RuleWithConstantAction(

Drop(Unspecified),

And(

Or(MessageCreateDmEvent, LastMessageReadUpdateDmEvent),

DmEventInOneToOneConversationWithUnavailableUser)) {

override def enableFailClosed = Seq(RuleParams.True)

}

object GroupEventInOneToOneConversationDropRule

extends RuleWithConstantAction(

Drop(Unspecified),

And(

Or(JoinConversationDmEvent, ConversationCreateDmEvent),

DmEventInOneToOneConversation)) {

override def enableFailClosed = Seq(RuleParams.True)

}

}