package com.twitter.home\_mixer.service

import com.twitter.conversions.DurationOps.\_

import com.twitter.product\_mixer.core.functional\_component.common.alert.Destination

import com.twitter.product\_mixer.core.functional\_component.common.alert.EmptyResponseRateAlert

import com.twitter.product\_mixer.core.functional\_component.common.alert.LatencyAlert

import com.twitter.product\_mixer.core.functional\_component.common.alert.NotificationGroup

import com.twitter.product\_mixer.core.functional\_component.common.alert.P99

import com.twitter.product\_mixer.core.functional\_component.common.alert.Percentile

import com.twitter.product\_mixer.core.functional\_component.common.alert.SuccessRateAlert

import com.twitter.product\_mixer.core.functional\_component.common.alert.predicate.TriggerIfAbove

import com.twitter.product\_mixer.core.functional\_component.common.alert.predicate.TriggerIfBelow

import com.twitter.product\_mixer.core.functional\_component.common.alert.predicate.TriggerIfLatencyAbove

import com.twitter.util.Duration

/\*\*

\* Notifications (email, pagerduty, etc) can be specific per-alert but it is common for multiple

\* products to share notification configuration.

\*/

object HomeMixerAlertConfig {

val DefaultNotificationGroup: NotificationGroup = NotificationGroup(

warn = Destination(emails = Seq("")),

critical = Destination(emails = Seq(""))

)

object BusinessHours {

val DefaultNotificationGroup: NotificationGroup = NotificationGroup(

warn = Destination(emails = Seq("")),

critical = Destination(emails =

Seq(""))

)

def defaultEmptyResponseRateAlert(warnThreshold: Double = 50, criticalThreshold: Double = 80) =

EmptyResponseRateAlert(

notificationGroup = DefaultNotificationGroup,

warnPredicate = TriggerIfAbove(warnThreshold),

criticalPredicate = TriggerIfAbove(criticalThreshold)

)

def defaultSuccessRateAlert(

threshold: Double = 99.5,

warnDatapointsPastThreshold: Int = 20,

criticalDatapointsPastThreshold: Int = 30,

duration: Int = 30

) = SuccessRateAlert(

notificationGroup = DefaultNotificationGroup,

warnPredicate = TriggerIfBelow(threshold, warnDatapointsPastThreshold, duration),

criticalPredicate = TriggerIfBelow(threshold, criticalDatapointsPastThreshold, duration),

)

def defaultLatencyAlert(

latencyThreshold: Duration = 200.millis,

warningDatapointsPastThreshold: Int = 15,

criticalDatapointsPastThreshold: Int = 30,

duration: Int = 30,

percentile: Percentile = P99

): LatencyAlert = LatencyAlert(

notificationGroup = DefaultNotificationGroup,

percentile = percentile,

warnPredicate =

TriggerIfLatencyAbove(latencyThreshold, warningDatapointsPastThreshold, duration),

criticalPredicate =

TriggerIfLatencyAbove(latencyThreshold, criticalDatapointsPastThreshold, duration)

)

}

}