package com.twitter.timelineranker.client

import com.twitter.conversions.DurationOps.\_

import com.twitter.finagle.builder.ClientBuilder

import com.twitter.finagle.mtls.authentication.EmptyServiceIdentifier

import com.twitter.finagle.mtls.authentication.ServiceIdentifier

import com.twitter.finagle.mtls.client.MtlsClientBuilder.\_

import com.twitter.finagle.param.OppTls

import com.twitter.finagle.service.RetryPolicy

import com.twitter.finagle.service.RetryPolicy.\_

import com.twitter.finagle.ssl.OpportunisticTls

import com.twitter.finagle.thrift.ThriftClientRequest

import com.twitter.servo.client.Environment.Local

import com.twitter.servo.client.Environment.Staging

import com.twitter.servo.client.Environment.Production

import com.twitter.servo.client.Environment

import com.twitter.servo.client.FinagleClientBuilder

import com.twitter.util.Try

import com.twitter.util.Duration

sealed trait TimelineRankerClientBuilderBase {

def DefaultName: String = "timelineranker"

def DefaultProdDest: String

def DefaultProdRequestTimeout: Duration = 2.seconds

def DefaultProdTimeout: Duration = 3.seconds

def DefaultProdRetryPolicy: RetryPolicy[Try[Nothing]] =

tries(2, TimeoutAndWriteExceptionsOnly orElse ChannelClosedExceptionsOnly)

def DefaultLocalTcpConnectTimeout: Duration = 1.second

def DefaultLocalConnectTimeout: Duration = 1.second

def DefaultLocalRetryPolicy: RetryPolicy[Try[Nothing]] = tries(2, TimeoutAndWriteExceptionsOnly)

def apply(

finagleClientBuilder: FinagleClientBuilder,

environment: Environment,

name: String = DefaultName,

serviceIdentifier: ServiceIdentifier = EmptyServiceIdentifier,

opportunisticTlsOpt: Option[OpportunisticTls.Level] = None,

): ClientBuilder.Complete[ThriftClientRequest, Array[Byte]] = {

val defaultBuilder = finagleClientBuilder.thriftMuxClientBuilder(name)

val destination = getDestOverride(environment)

val partialClient = environment match {

case Production | Staging =>

defaultBuilder

.requestTimeout(DefaultProdRequestTimeout)

.timeout(DefaultProdTimeout)

.retryPolicy(DefaultProdRetryPolicy)

.daemon(daemonize = true)

.dest(destination)

.mutualTls(serviceIdentifier)

case Local =>

defaultBuilder

.tcpConnectTimeout(DefaultLocalTcpConnectTimeout)

.connectTimeout(DefaultLocalConnectTimeout)

.retryPolicy(DefaultLocalRetryPolicy)

.failFast(enabled = false)

.daemon(daemonize = false)

.dest(destination)

.mutualTls(serviceIdentifier)

}

opportunisticTlsOpt match {

case Some(\_) =>

val opportunisticTlsParam = OppTls(level = opportunisticTlsOpt)

partialClient

.configured(opportunisticTlsParam)

case None => partialClient

}

}

private def getDestOverride(environment: Environment): String = {

val defaultDest = DefaultProdDest

environment match {

// Allow overriding the target TimelineRanker instance in staging.

// This is typically useful for redline testing of TimelineRanker.

case Staging =>

sys.props.getOrElse("target.timelineranker.instance", defaultDest)

case \_ =>

defaultDest

}

}

}

object TimelineRankerClientBuilder extends TimelineRankerClientBuilderBase {

override def DefaultProdDest: String = "/s/timelineranker/timelineranker"

}