package com.twitter.product\_mixer.shared\_library.manhattan\_client

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

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

import com.twitter.finagle.ssl.OpportunisticTls

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.manhattan.v1.{thriftscala => mh}

import com.twitter.storage.client.manhattan.kv.Experiments

import com.twitter.storage.client.manhattan.kv.Experiments.Experiment

import com.twitter.storage.client.manhattan.kv.Guarantee

import com.twitter.storage.client.manhattan.kv.ManhattanKVClient

import com.twitter.storage.client.manhattan.kv.ManhattanKVClientMtlsParams

import com.twitter.storage.client.manhattan.kv.ManhattanKVEndpoint

import com.twitter.storage.client.manhattan.kv.ManhattanKVEndpointBuilder

import com.twitter.storage.client.manhattan.kv.NoMtlsParams

import com.twitter.storehaus\_internal.manhattan.ManhattanCluster

import com.twitter.util.Duration

object ManhattanClientBuilder {

/\*\*

\* Build a ManhattanKVClient/Endpoint [[ManhattanKVEndpoint]] / [[ManhattanKVClient]]

\*

\* @param cluster Manhattan cluster

\* @param appId Manhattan appid

\* @param numTries Max number of times to try

\* @param maxTimeout Max request timeout

\* @param maxItemsPerRequest Max items per request

\* @param guarantee Consistency guarantee

\* @param serviceIdentifier Service ID used to S2S Auth

\* @param statsReceiver Stats

\* @param experiments MH client experiments to include

\* @return ManhattanKVEndpoint

\*/

def buildManhattanEndpoint(

cluster: ManhattanCluster,

appId: String,

numTries: Int,

maxTimeout: Duration,

guarantee: Guarantee,

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver,

maxItemsPerRequest: Int = 100,

experiments: Seq[Experiment] = Seq(Experiments.ApertureLoadBalancer)

): ManhattanKVEndpoint = {

val client = buildManhattanClient(

cluster,

appId,

serviceIdentifier,

experiments

)

ManhattanKVEndpointBuilder(client)

.defaultGuarantee(guarantee)

.defaultMaxTimeout(maxTimeout)

.maxRetryCount(numTries)

.maxItemsPerRequest(maxItemsPerRequest)

.statsReceiver(statsReceiver)

.build()

}

/\*\*

\* Build a ManhattanKVClient

\*

\* @param cluster Manhattan cluster

\* @param appId Manhattan appid

\* @param serviceIdentifier Service ID used to S2S Auth

\* @param experiments MH client experiments to include

\*

\* @return ManhattanKVClient

\*/

def buildManhattanClient(

cluster: ManhattanCluster,

appId: String,

serviceIdentifier: ServiceIdentifier,

experiments: Seq[Experiment] = Seq(Experiments.ApertureLoadBalancer)

): ManhattanKVClient = {

val mtlsParams = serviceIdentifier match {

case EmptyServiceIdentifier => NoMtlsParams

case serviceIdentifier =>

ManhattanKVClientMtlsParams(

serviceIdentifier = serviceIdentifier,

opportunisticTls = OpportunisticTls.Required)

}

val label = s"manhattan/${cluster.prefix}"

new ManhattanKVClient(

appId = appId,

dest = cluster.wilyName,

mtlsParams = mtlsParams,

label = label,

experiments = experiments

)

}

def buildManhattanV1FinagleClient(

cluster: ManhattanCluster,

serviceIdentifier: ServiceIdentifier,

experiments: Seq[Experiment] = Seq(Experiments.ApertureLoadBalancer)

): mh.ManhattanCoordinator.MethodPerEndpoint = {

val mtlsParams = serviceIdentifier match {

case EmptyServiceIdentifier => NoMtlsParams

case serviceIdentifier =>

ManhattanKVClientMtlsParams(

serviceIdentifier = serviceIdentifier,

opportunisticTls = OpportunisticTls.Required)

}

val label = s"manhattan/${cluster.prefix}"

Experiments

.clientWithExperiments(experiments, mtlsParams)

.build[mh.ManhattanCoordinator.MethodPerEndpoint](cluster.wilyName, label)

}

}