ViPR S3 Client for Java

Getting Started

The ViPR S3 client extends Amazon’s own S3 client implementation included in the AWS SDK for Java. As such, it depends on the AWS SDK library and all of its transitive dependencies.

# Enhancements

The ViPR S3 client includes the following ViPR-specific enhancements to the standard S3 API published by Amazon. If you wish to leverage any of these enhancements, you will need to use the ViPR S3 SDK. If none of these enhancements apply to you, you may consider using the standard AWS SDK instead. If you isolate the client creation appropriately, the code you write using the AWS SDK will be compatible should you choose to add these features later.

All of the below features are described in more detail in the ViPR S3 Programmer’s Guide - ??.

## Object Modification

ViPR allows object modification by including ranges in your PUT requests. You can modify any part of an object, append to the end of an object or even create a “sparse” object by writing past the end of its existing data size (bytes in between will be filled with zeros - ??).

## Atomic Append

There may be cases where you need to append to an object, but determining the exact byte offset is inefficient and isn’t vital to your use-case. For this scenario, ViPR provides the ability to atomically append data to the object without specifying an offset (the correct offset is returned to you in the response).

## Bucket File Access

Using ViPR, you may optionally toggle the access mode of a bucket between the standard REST access (default) and filesystem access (via NFS) using the file access feature. When filesystem access is enabled for a bucket, all existing objects in the bucket are no longer accessible via REST, but are mountable using an NFS client. Mount points for each object can be obtained with a GET call.

## Namespaces

ViPR provides namespaces that isolate an entire bucket/key space. A bucket/key in one namespace is not accessible and may even represent a completely different object in another namespace. You can optionally specify a namespace via DNS hostname. If you do not use DNS to specify the namespace for a request, you will need to use the ViPR S3 client and call .setNamespace() before interacting with objects.

# Maven Setup

If you project uses maven or ivy for dependency management, then adding the ViPR S3 Client is quite easy. Simply include the following dependency in your pom.xml file:

<dependency>

<groupId>com.emc</groupId>

<artifactId>vipr-data-services-sdk</artifactId>

<version>1.0</version>

</dependency>

# Manual Setup

If you are manually managing your dependencies, then you will need to download the following:

## Amazon AWS SDK for Java

http://sdk-for-java.amazonwebservices.com/latest/aws-java-sdk.zip

## ViPR Data Services SDK

http://search.maven.org/remotecontent?filepath=com/emc/vipr-data-services-sdk/1.0.0/vipr-data-services-sdk-1.0.0.jar

You will need the following jars on your project classpath for all scopes (versions are accurate as of this writing):

* aws-sdk-java-1.4.6.jar
* commons-codec-1.3.jar
* commons-logging-1.1.1.jar
* httpclient-4.1.jar
* httpcore-4.1.jar
* jackson-core-asl-1.8.9.jar
* jackson-mapper-asl-1.8.9.jar
* hamcrest-core-1.3.jar

# Configuration

To instantiate and configure the ViPR S3 client, you simply use the ViPRS3Client constructor and pass in your BasicAWSCredentials as you would with the AmazonS3Client. The only additional required step is to specify your endpoint (the ViPR data services node or load-balancer). You may optionally specify a namespace by calling setNamespace if you are not using DNS-based namespaces.

BasicAWSCredentials creds = new BasicAWSCredentials(accessKey, secretKey);

ViPRS3Client client = new ViPRS3Client(creds);

client.setEndpoint(endpoint);

// if not using DNS-based namespace

client.setNamespace(namespace);

The client extends AmazonS3Client and implements both the ViPRS3 and AmazonS3 interfaces, so any existing S3 code will continue to work with this instance.

## Conditionalizing ViPR-specific calls

In cases where you can’t anticipate your target environment, or where you must execute against both S3 and ViPR, you can branch your logic based on whether you are provided a ViPR client or not.

if (client instanceof ViPRS3) {

((ViPRS3) client).updateObject(bucket, key, inputStream,

metadata, startOffset);

} else {

// recreate new complete object

client.putObject(bucket, key, inputStream, metadata);

}

# Sample Project

A sample Eclipse project is provided to demonstrate all of the ViPR enhancements with working examples. You can download the project here:

<https://github.com/emcvipr/dataservices-s3-java/sample-project.zip>