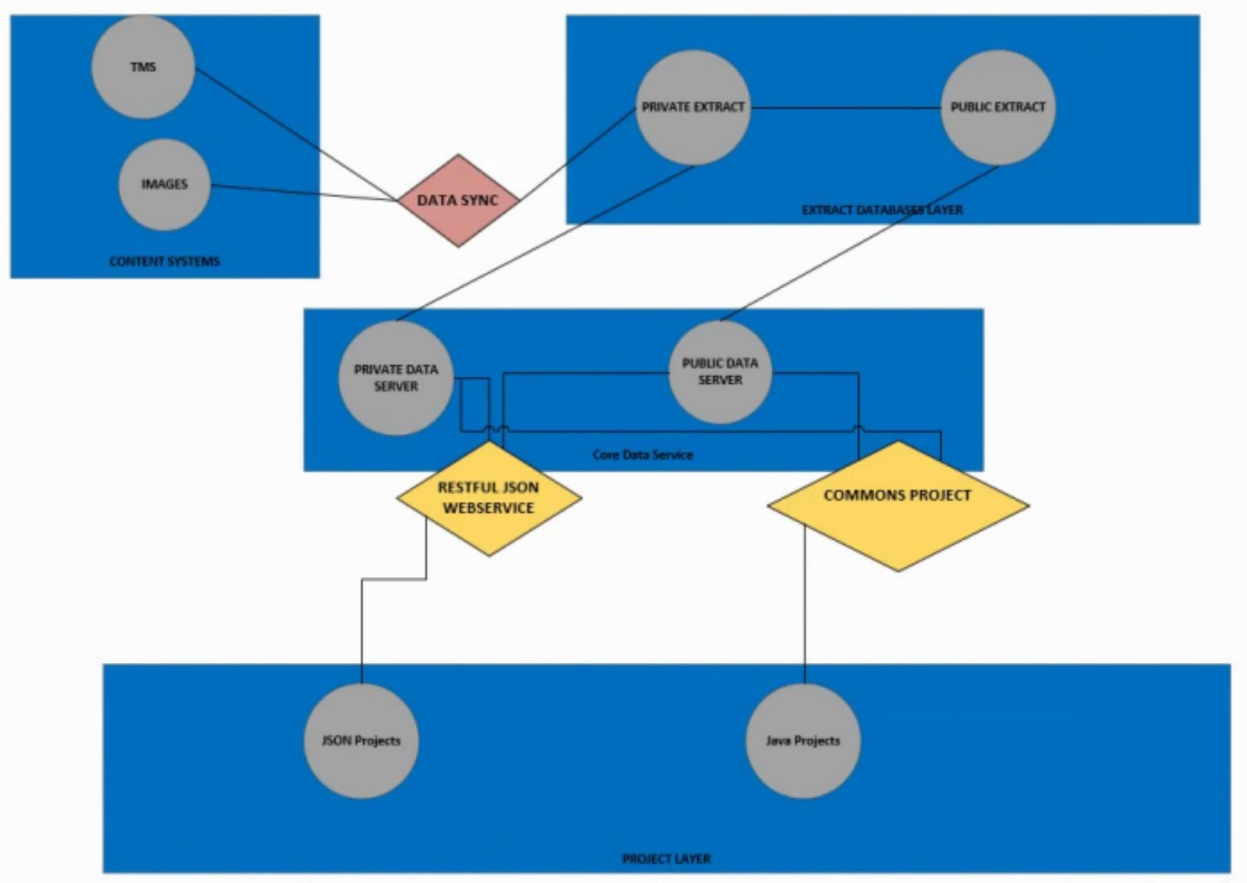
Using Core Data Service via Java APIs

Overview

TTS has been progressing over the pass few years in centralizing the modeling of the modeling of the TMS data structures. This has resulted in the Java classes the make up the core of the Data Sever, Web Redesign, and Commons API projects. But as the multiple projects mentioned previously would indicate, further work needed to be done in order to both centralize the data modeling and the code base for accessing the actual data.

The next step in this work has been achieved with the Core Data Service (CDS) project. This project brings together the Centralizing of the Data Modeling (Commons Project) with the centralizing of the data repository through support for multiple types of Remote Procedure Calls. This document focuses on an overview of how Core Data Service is used through Java APIs.

Design



The above diagram is an illustration of the current architecture for the Core Date Service. As illustrated, The Commons Project is responsible for defining the Date Models as well as supplying access for Java project retrieving TMS data from the Core Data Service.

The Commons Project

The Commons Project originated as an AEM specific Java project. Starting from Version 2.1, The Commons Project is now a stand alone JAR file that can be built/installed from Maven.

```
<dependency>
    <groupId>gov.nga</groupId>
    <artifactId>common</artifactId>
    <version>2.1.0-SNAPSHOT</version>
</dependency>
```

Configuration

```
1 package gov.nga.common.rpc.client;
2
3 public interface ClientConfiguration
4 {
5     public String getConnectionURL();
6 }
7
```

Configuration for CDS is handled by the ClientConfiguration Interface. Each Java project must create its own Object instance of this configuration. Currently, the interface only has 1 method "getConnectionURL()". This is expected to return the URL (including port) for CDS. The table below displays the current URLs for CDS.

Server Name	URL
Test	https://ap-artdataservicetst-priv.nga.gov:9090
Production Private Data	https://ap-artdataservice-priv.nga.gov:9090

Connection API

Actual connection with CDS is handled by the "DataServiceManager" (gov.nga.common.rpc.DataServiceManager) Singelton class.

```
import gov.nga.common.rpc.DataServiceManager;
import gov.nga.common.rpc.client.ClientConfiguration;

public class ExhibitionTestingClient
{
    final private DataServiceManager manager;

    public ExhibitionTestingClient(final ClientConfiguration config)
    {
        manager = DataServiceManager.getManager(config);
    }

    public void runTest()
```

The DataServiceManager is comprised of two Service Interfaces

- ArtDataQuerier (gov.nga.common.entities.art)
  This interface is a continuation from version 1 of The Commons Project. These are the principal

methods for retrieving TMS Data Objects.

- DataServiceQuerier (gov.nga.common.rpc)
  This interface extends ArtDataQuerier and contains methods meant to replace the role that the local DataManager and Cacher served in version 1 of The Commons Project.

```
public interface DataServiceQuerier extends ArtDataQuerier
{

    public Date getLastSyncTime();

    public Collection<Long> getAllArtObjectIDs();
    public Collection<Long> getAllExhibitionIDs();
    public Collection<Long> getAllConstituentIDs();
}
```

We will focus a little more now on the methods in DataServiceQuerier.

- **getLastSyncTime()**
  In the future, CDS should be able to real time communicate with subscribers about events like when the data cache updates. But for now, the responsibility is on the Java Client if it has routines dependent on refreshing when the cache does. This method will return a Date object with the timestamp of the last CDS cache sync. CDS generally syncs once a day, but it is probably good practice for Java projects to check this method a few times a day in case the cache updates due to an off schedule task.

- **getAll….IDs()**
  For Java Projects that need to do their own data collection work, these methods take the place of the ability to call the ArtDataCacher directly in version 1 of The Commons Project. These methods only return ids which allow each Java Project to manage data object retrieval from both a memory and network performance perspective. There is a corresponding fetchBy…IDs method in the ArtDataQuerier for each of these data types. One recommended approach is to break the id list into segments of 25 – 40 ids at a time to send to the fetch routines (this can even be threaded in order to increase performance).