# General REST API Access for DDMAP

**AWS Glue Catalog** stores technical metadata about Tables and their Columns. This metadata is essentially the same information that comprises basic **RAML Data Types**. **RAML Data Types** describe the data in an API.

<https://github.com/raml-org/raml-spec/blob/master/versions/raml-10/raml-10.md#raml-data-types>

Existing **Common Glue Functions** library includes a function to transform catalog data into basic **RAML Data Type** definitions. Output from this function can be saved to a file or copy/pasted to a text editor:

**def export\_glue\_to\_raml( glue\_database\_name, glue\_table\_name, output\_path='' ):**

**''' Export Glue table metadata to Mule-compatible RAMLv1.0 '''**

(This **Common Glue Functions** library was created to support DDMAP's Tableau Export, but there are no dependencies on other DDMAP functionality. So, this will work with ANY account's Glue Catalog, potentially including Glue Catalogs shared among accounts.

<https://bitbucket.devops.treasury.gov/projects/DUDE/repos/dude-file-processing-4/browse/common/glue_functions/python/glue_functions.py#193> )

**RAML Data Type** definitions can be published (uploaded) as **API Spec Fragments** on **Mule Anypoint Exchange** whence they can be discovered and reused (by reference) within multiple API specifications. In addition to data types, **API Spec Fragments** can also be used to establish other types of **building blocks** for functionality and additional characteristics.

[Overview of API Fragments Design Strategies | MuleSoft Blog](https://blogs.mulesoft.com/api-integration/patterns/api-fragments-design-strategies/)

<https://blogs.mulesoft.com/api-integration/patterns/api-fragments-design-strategies/>

Common API functionality should include a building block (GET method) to retrieve all rows from a table identified by any specified **RAML Data Type** definition.

e.g., API call …

https://{base\_url}/api/table-retrieve-all/{TableName}

… to run a **Mule Anypoint flow** (program) to execute the following SQL statement on the Glue Table

select \* from {TableName}

The Mule flow can utilize a [Mule Database Connector](https://docs.mulesoft.com/db-connector/1.13/) to execute SQL via JDBC and pass the result set to subsequent flow processing. Instead, I suggest passing the SQL to a Lambda function via the [Mule Amazon Lambda Connector](https://docs.mulesoft.com/amazon-lambda-connector/1.0/). The JDBC connector requires network ACLs to the RDBMS port, and database credentials configured/maintained in Mule. The Lambda connector goes thru standard already-accessible HTTPS endpoints, and enables database credentials to be maintained on the target system. Lambda also can support enhanced backend processing, such as including metadata with the result set, output format options (json vs csv), file output vs. stream, async processing, etc.

So far, this write-up has described how to make an entire table accessible via the Bureau's **Enterprise Anypoint API platform**. Beyond this basic requirement, it would be nice to support parameters which enable us to select a subset of table rows and columns, sort, paginate, and perhaps aggregate the result set. Such functionality is currently supported on the FiscalData web site as described at this link …

<https://fiscaldata.treasury.gov/api-documentation/#parameters>

e.g., API call …

https://{base\_url}/api/{TableName}?fields=**{ColumnList}**&filter=**{FilterSpec}**&sort=**{SortSpec}**&page[number]={**PageNumber**}&page[size]={**PageSize**}&format={**'json|csv'**}

… gets transformed to SQL statement

select **{ColumnList}**

from **{TableName}**

where {**FilterSpec**}

order by {**SortSpec**}

limit {**PageSize**} offset {**PageSize \* PageNumber**} ;

SQL Execution via Mule DB Connector or Lambda, as described above. **&format** specification is applied to the result set.

Offset Pagination (instead of Cursor Pagination) will be sufficient for this service. Huge data requests, too large to be offset-paged practically, can be output to S3 whence they can be shared, transferred, or downloaded as files.

<https://stackoverflow.com/questions/55744926/offset-pagination-vs-cursor-pagination>

<https://blogs.mulesoft.com/dev-guides/api-design/api-pagination-patterns/>

FRB has suggested that the 'more expedient' way to support this enhanced functionality is to copy and modify the DTAS custom application that implements FiscalData. This would require additional EC2 instance(s) to be built, operated, and maintained, as well as establishing network connectivity from Mule to APIs hosted on them. The DTAS app stores metadata in database tables, so presumably would need RDS. Then the application would need modified to support Glue and RAML.

I would rather see us use our time to implement a solution that is more compliant with Enterprise API standards, without introducing additional moving parts. I suggest the following tasks:

1. Develop a Lambda function that implements **export\_glue\_to\_raml** and front-end it with a Mule API that uses the Mule Lambda connector.
   * Develop and test on workstation using AnypointStudio.
   * Publish the API in the Training environment on the Bureau's Anypoint Exchange and test access from the Internet.

1. Use API created in Task 1 to Create API Spec Fragments on FS Anypoint Exchange for each of the datasets exported daily from Tableau (TDSX)

1. Develop a **SQL\_Exec** Lambda function that executes SQL query against Redshift and front-end it with a Mule API
   * Data access authorization based on Redshift credential stored in SSM
   * Supports S3 Query-in-place via Redshift Spectrum
   * Basic synchronous SQL execution / return result set as either CSV or JSON
     + As enhancement, consider [Using the Amazon Redshift Data API](https://docs.aws.amazon.com/redshift/latest/mgmt/data-api.html) as asynchronous execution option (for result sets that take longer than Lambda limit (15 min) to process)
     + Returns Redshift error if SQL statement is invalid
   * Returns JSON execution metadata
     + RAML Data Type of result set
     + Record count(s)
     + SQL Statement executed
     + Etc ?

1. Create a Mule API application that references API Spec Fragments (Task 2) which calls Lambda (Task 3) to retrieve any of the TDSX tables
   * e.g., API call …

https://{base\_url}/api/table-retrieve-all/{TableName}

1. Develop a **URL\_to\_SQL** Lambda function that transforms URL parameters into a SQL statement and front-end it with a Mule API
   * URL parms similar to FiscalData specification:

<https://fiscaldata.treasury.gov/api-documentation/#parameters>

* Package **def transform\_URL\_to\_SQL ( url\_input )** into common library
  + implemented as a Lambda layer, and also can be sourced elsewhere (e.g., SageMaker, PySpark)
  + Enhance [RedshiftDb custom class](https://bitbucket.devops.treasury.gov/projects/DUDE/repos/dude-file-processing-4/browse/common/psycopg2-3.7/python/psycopg2/redshift.py) or incorporate into [Glue Common Functions](https://bitbucket.devops.treasury.gov/projects/DUDE/repos/dude-file-processing-4/browse/common/glue_functions/python/glue_functions.py)
* Returns JSON
  + SQL Statement parsed from URL parms (or simply 'select \* from {TableName}' if no parms)
  + RAML Data Type describing {ColumnList} using attributes from Glue Catalog

1. Enhance Mule API application (Task 4), insert flow to call **URL\_to\_SQL** (Task 5), prior to invoking **SQL\_Exec** (Task 3) passing SQL transformed from URL