REST API Integration - Query Database View

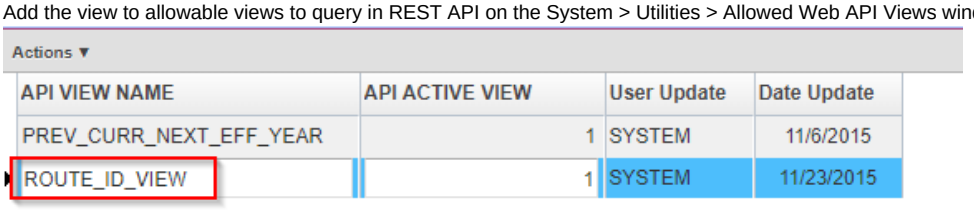
Complex data joins can be exposed from the AgileAssets system using database views and the REST API.

The following examples shows the process.

1. First create the view in the database.
   e.g. Create a view ROUTE_ID_VIEW

```
CREATE OR REPLACE FORCE VIEW ROUTE_ID_VIEW ("ROUTE_ID") AS
SELECT DISTINCT ROUTE_ID
FROM SETUP_NETWORK_LINES
ORDER BY ROUTE_ID;
```

2. Add the view to allowable views to query in REST API on the System > Utilities > Allowed Web API Views window.



3. Get an access token to use for REST API request - see REST API V2 Security document for detail.
4. Use the access token to query the rest endpoint using a GET HTTP request
   https://akdot.agileassets.com/AMS_AK_DEV/rest/v1/lookup/view/ROUTE_ID_VIEW
5. The following JSON is returned which is the values from the database view

```
[
{
"ORACLEROWNUM": 1,
"ROUTE_ID": 74523
},
{
"ORACLEROWNUM": 2,
"ROUTE_ID": 74524
},
{
"ORACLEROWNUM": 3,
"ROUTE_ID": 74525
},
{ ....
```

   Note: By default only the first 200 records are returned.
6. Use the "q" parameter to return more records
   e.g. to return the first 100 records
   https://akdot.agileassets.com/AMS_AK_DEV/rest/v1/lookup/view/ROUTE_ID_VIEW?q={"page":{"size":1000,"number":1}}

JavaScript Example

```
var data = null;

var xhr = new XMLHttpRequest();
xhr.withCredentials = true;

xhr.addEventListener("readystatechange", function () {
if (this.readyState === 4) {
console.log(this.responseText);
}
});

xhr.open("GET", "https://akdot.agileassets.com/AMS_AK_DEV/rest/v1/lookup/view/ROUTE_ID_VIEW?q={%22page%22:{%
22size%22:1000,%22number%22:1}}");
xhr.setRequestHeader("Authorization", "Bearer $2a$12$YT4WGAVMaSP09.qisYMp6OP/VpAUtynPQUqIRtTm9dU6A3lfWMKRW");
xhr.setRequestHeader("Cache-Control", "no-cache");

xhr.send(data);
```

The "Q" Object Reference

The "Q" object is optional object of JSONObject type that can be passed as @QueryParameter into many API calls. Purpose of this object is to format response of API response.
Example of such object:

```
{
"projection":["WORK_ORDER_ID","STATUS"],
"filter":[["STATUS","=",4], ["OWNER_ID","IN","1317,56"], "AND"],
"sort":{"START_DATE":"DESC"},
"page":{"size":20, "number":2}
}
```

All fields of object are optional and have the following meaning:

- **projection** = JSON array of column names to be returned by request. In example above only [WORK_ORDER_ID] and [STATUS] columns would be return. If that field is omitted the request would return ALL available columns.
    - This can viewed as VERTICAL filtering.
- **sort** = JSON object, where keys are column names and values are direction of sort: DESC or ASC. This field specifies that output should be sorted.
- **filter** = JSON array that made of 3 element JSON arrays and strings "AND" or "OR".
    - The inner JSON arrays represent condition to filter by.
        - First element is operand, usually a column name
        - Second element is operator, such as "=", "BETWEEN", "IN"
        - Third element is operand to compare with, usually some value
    - The "AND" or "OR" strings tell how conditions should be related in the WHERE clause.
    - NOTE: the filter is written using RPN(Reverse Polish Notation) where
        - Operators : "AND", "OR"
        - Operands : 3-element JSON condition arrays.
    - This can viewed as HORIZONTAL filtering.
- **page** = represents pagination of the output.
    - size = number of records per page, first record has index 1.
    - number = page number, first page has index 1.

In the following example we request all work orders, and with the "q" object limit response to return only open work orders (Status = 4). The response contains only two columns, sorted by work order start date and returns second page where there are 20 rows per page:
<baseUrl>/workOrder?q={"filter":[["STATUS","=",4]],"projection":["WORK_ORDER_ID","STATUS"],"sort":{"START_DATE":"DESC"},"page":{"size":20,"number":2}}

Note: *When combining the projection with the sort & filter features, columns referenced in the sort and filter parameters* ***must*** *be included in the projection parameter.*
Note*: When using a default q filter a default of 200 maximum records are returned. Use the* ***page*** *field to set the "size" to a high value (number of records to return) and set "number" to 1.*