Query Service in vCloud Director

This document covers a special API which would be helpful for vRO developers who are working in complex multi-tenant vCD based cloud platforms but even in small environments, dramatic latency difference can be noticed.

Long ago, VMware vCloud Director 1.5 introduced a VMware vCloud API query service, which can significantly improve developer efficiency, by minimizing the number of API requests and the amount of data transferred for an API client to obtain needed information. Example query parameters include **sorting and ordering, pagination, filtering, projection,** and **expressions.**

This article explains the reasons you may have, to leverage the query service and gives an example on how to use it.

- First you may wonder why you should use it. The short answer is because it is more efficient for finding objects based on their properties.
- The longer answer is that with using the vCloud API: You need to drill down through the hierarchy of objects to find a particular object and its properties and need to handle things such as sorting and filtering in your code.

If you need to get to a list of objects such as vApps your vCloud API client will issue an HTTP GET per object. Each vApp XML object representation can be a few KBytes with its OVF envelope. Multiply this by thousands of objects and you can imagine how many requests and how much data can be transferred just to get a simple information for a few vApps you want to filter on a certain property.

The query service is done server side. Your client sends a simple query which is executed on the server and sending back only the requested, filtered information. The information is sent in pages, meaning that you can at any time stop to get the next pages if you have the information needed.

Using the query service means fewer complex operations for the developer, and much better performance for getting the information.

Real Life vRO Scenario

Now let's see a practical example using the query service with vCO.

I needed to delete a vApp Template but then I needed to remove it from the catalogs it was in. If you are familiar with the vCloud workflow library you may know there is a "Delete Catalog Item and linked item". This is fine when you have a catalog item you want to delete with its vApp Template but it does not work the other way around.

First solution: Using the vCloud API

```
var catalogItemsOut = new Array();

var org = vAppTemplate.parent.parent;

var catalogs = org.getCatalogs()


for each (var catalog in catalogs) {

    var catalogItems =  catalog.getCatalogItems();

    for each (var catalogItem in catalogItems) {

        if (catalogItem.entity.href ==
vAppTemplate.getReference().href) {

            System.log(catalogItem.name + " : " +
catalogItem.entity.href);

            catalogItemsOut.push(catalogItem);

        }

    }

}

return catalogItemsOut;
```

From the vAppTemplate getting the parent VDC, then the parent Organization. From there getting all the catalogs, all their catalog items and check if they reference the vApp Template.

Second solution: Using the query service

```
var catalogItems = new Array();

var vcdHost = vAppTemplate.getHost();

var queryService = vcdHost.getQueryService();

var expression = new
VclExpression(VclQueryCatalogItemField.ENTITY,
vAppTemplate.getReference().href , VclExpressionType.EQUALS);

var filter = new VclFilter(expression);

var params = new VclQueryParams();

params.setFilter(filter);

var resultSet =
queryService.queryRecords(VclQueryRecordType.CATALOGITEM,
params);

while (resultSet != null)  {

    var records = resultSet.getRecords(new
VclQueryResultCatalogItemRecord());

    for each (var record in records) {

        var catalogItemRef = new VclReference();

        catalogItemRef.href = record.href;

        catalogItemRef.name = record.name;

        catalogItemRef.type = record.type;

          var catalogItem =
(vcdHost.getEntityByReference(VclFinderType.CATALOG_ITEM,
catalogItemRef));

        catalogItems.push(catalogItem);

    }

    resultSet = resultSet.getNextPage();
}
return catalogItems;
```

First get the vCloud Director host from the vApp Template, then create an expression looking for the catalog item entity having an HREF equals the one of the vAppTemplate. For each record create a reference to the catalog item, get the object form the reference and add it to the array of catalogItems.

x7 faster

I have a small remote vCloud Director Cloud test environment with two catalogs and a total of 16 catalogs. Does using the query service makes a difference in such a small environment?

Running a workflow looking for the same Template using the two solutions it takes:

About 7 seconds and about 400 lines of XML to get all the objects with the vCloud API

(1 VDC, 1 Organization, catalogs and their catalog items)

About 1 second and 12 lines of XML (QueryResultRecords)

To see the XML exchanged the Debug mode is activated on the vCloud Director plug-in (as described in this article)

The catalog and catalog items objects are a lot smaller than objects such as the vApps so you can imagine how much difference it can do when querying in an environment with thousands of vApps.

Who's the winner, vCloud API or Query Service?

Now you may think the clear winner is always the Query service but **it is not always the case**. If you run the workflow using the vCloud API a second time it runs in less than a second and no XML in the logs. In fact, the vCloud Director plug-in cache is coming to the rescue. The objects are already in the vCO server allocated memory and no exchange need to be done with the vCloud Server. This is very useful but it has a couple of drawbacks:

The heavy lifting of getting the necessary objects must have been done at least once in the last 30 minutes (cache TTL).

The objects state does not get update from the vCloud server until you refresh these objects, which mean downloading them again using the updateInternalState() method. For example, any vCD library workflow will update the changed objects but other calls to the vCD server won't (vCloud Director UI or API).