Sample REST API Calls and Use Case Descriptions

Create a Category

Verb: POST
Request URL: https://9.148.55.217:9445/ibm/iis/igc-rest/v1/assets/

Body:
{
"_type" : "category",
"short_description" : "new category",
"name" : "cat evo"
}

Response Code: 201
Response Header:
{
"X-Powered-By": "Servlet/3.0",
"Cache-Control": "no-cache",
"Pragma": "no-cache",
"Expires": "Thu, 01 Jan 1970 00:00:00 GMT",
"X-Frame-Options": "SAMEORIGIN",
"Location": "https://9.148.55.217:9445/ibm/iis/igc-rest/v1/assets/6662c0f2.ee6a64fe.6vcktlbm1.fsnfb6d.apv9ds.8vk9kutt1evmk4iik23vm",
"Content-Language": "en-US",
"Content-Length": "0",
"Date": "Tue, 28 Jul 2015 16:19:19 GMT"
}

Get Details of an Asset (in this case, the new category we just created above)

Verb: GET
Sample URL: https://9.148.55.217:9445/ibm/iis/igc-rest/v1/assets/6662c0f2.ee6a64fe.6vcktlbm1.fsnfb6d.apv9ds.8vk9kutt1evmk4iik23vm
Body:
{
"created_by": "Administrator IIS",
"modified_on": "2015-07-28T19:19:20",
"_type": "category",
"short_description": "new category",
"_id": "6662c0f2.ee6a64fe.6vcktlbm1.fsnfb6d.apv9ds.8vk9kutt1evmk4iik23vm",
"modified_by": "Administrator IIS",
"_context": [],
"created_on": "2015-07-28T19:19:20",
"_url": "https://9.148.55.217:9445/ibm/iis/igc-rest/v1/assets/6662c0f2.ee6a64fe.6vcktlbm1.fsnfb6d.apv9ds.8vk9kutt1evmk4iik23vm",
"name": "cat evo",
"_name": "cat evo"
}
Response Code: 200

Create a Term Within a Category

Verb: POST
Sample URL: https://9.148.55.217:9445/ibm/iis/igc-rest/v1/assets/

Body:
{
"_type" : "term",
"short_description" : "new term",
"name" : "term evo 01",
"status":"ACCEPTED",
"parent_category":"6662c0f2.ee6a64fe.6vcktlbm1.fsnfb6d.apv9ds.8vk9kutt1evmk4iik23vm"
}

Response Code: 201
Response Header: (new RID matches RID in the "Update a Term" section described next)
{
"X-Powered-By": "Servlet/3.0",
"Cache-Control": "no-cache",
"Pragma": "no-cache",
"Expires": "Thu, 01 Jan 1970 00:00:00 GMT",
"X-Frame-Options": "SAMEORIGIN",
"Location": "https://9.148.55.217:9445/ibm/iis/igc-rest/v1/assets/6662c0f2.e1b1ec6c.6vcktlc33.ofq9lnj.40o7b7.shouk2meamc84dm7c3aq8",
"Content-Language": "en-US",
"Content-Length": "0",
"Date": "Tue, 28 Jul 2015 16:33:36 GMT"
}

Update a Term (the same one we just created)

Verb: PUT
Sample URL: https://9.148.55.217:9445/ibm/iis/igc-rest/v1/assets/6662c0f2.e1b1ec6c.6vcktlc33.ofq9lnj.40o7b7.shouk2meamc84dm7c3aq8

Body:
{
"long_description":"new long desc2",
"example":"example X",
"abbreviation":"abbreviation X",
"usage":"usage X"
}

Add a Label to a Term (the same to add a label to ANY asset)

Verb: PUT
Sample URL: https://win764-vmware:9443/ibm/iis/igc-rest/v1/assets/6662c0f2.e1b1ec6c.7pm4rcajd.a36jlvd.dviqti.n8eqmjtbsbg76t59mhujr

Body:

```
{
 "labels": {
   "items": [
     {
      "_id": "6662c0f2.22257cc4.7pm4na6go.a1142ak.iq0rs2.jheloapd50afp2i0qc9sa"
     }
] }}
```


Update a Basic String Custom Attribute for an Asset

Verb: PUT
Sample URL: https://win764-vmware:9443/ibm/iis/igc-rest/v1/assets/b1c497ce.60641b50.7pm4o13vv.ea06vgu.l5kdj0.f2tfnn7v7enart4ati7cn

Body: (constructed dynamically after issuing the first GET)**
{

"custom_IT Business Element Name":"sample short description", prepend custom_ to the custom attribute name
"custom_IT Business Element Description":"sample long description"
}
** Use the **GET/types/** calls to find the names for the custom attributes.

Assign Assets to a Term and Construct the JSON body

What should the JSON code in the Body look like for any action that you want to perform?

Follow these steps:

In InfoSphere Information Governance Catalog, do these steps:

1. Create a term or asset that has what you want. For example, if you want to learn how to assign assets, then create a term and assign some assets by using the user interface.
2. Get the RID of the new asset from the URL by doing these steps:
   a) Open the Details page of the new asset.
   b) Copy the URL, and then paste it into some text editor. For example, the URL with the RID in yellow highlight might be https://localhost:9443/ibm/iis/igc/#dossierView/6662c0f2.e1b1ec6c.7pm4rcajd.a36jlvd.qb10ki.gpsefbgv0v3av6kboiev?bg_req_context=%7B%22perspective%22%3A%22PublishedGlossary%22%7D.

In InfoSphere Information Governance Catalog REST API, do these steps:

1. Expand **Assets,** and then click **/GET /assets/{id}**
2. Type the RID of the new asset into the ID field.
3. Now review the JSON response. In this example, look for assigned assets:
4. Copy and paste that JSON text into some text editor. Remove everything except the RIDs. The result might be like this JSON snippet :


```
{"assigned_assets": {

   "items": [
    {
          "_id": "b1c497ce.6e83759b.7pm4o13vt.t2allb0.m95ris.skmo4va1v71vfi7pqctlq"
        },
    {
"_id": "b1c497ce.6e83759b.7pm4o130p.89ei8iu.ftq0qm.09svmjl0vtmp02dkrb4hf"
}
  ]}}
```


5. Replace those RIDs with the ones that are important, and then paste this JSON code into the String field of the **PUT /assets/{id}** command. Use the RID of the term that you want to actually add these assets to:

PUT /assets/{id}

Implementation Notes

Update an asset by passing in a JSON object with the updated properties.

See the 'Show Examples' section to see the syntax of the JSON object.

Show Examples

Parameters

Parameter	Value	Description	Parameter Type
id	krcajd.a36jlvd.dviqti.n8eqmjtbsbg76t59mhujr	**The RID of the asset**	path
string	{"assigned_assets": { "items": [{ "_id": "b1c497ce.6e83759b.7pm4o13vt.t2allb Parameter content type: application/json	**JSON Object representing the properties of the asset to update.**	body

Response Messages

HTTP Status Code	Reason	Response Model
401	Unauthorized	

The final API looks like this:

Verb: **PUT**
Sample URL: https://win764-vmware:9443/ibm/iis/igc-rest/v1/assets/6662c0f2.e1b1ec6c.7pm4rcajd.a36jlvd.dviqti.n8eqmjtbsbg76t59mhujr

Body:

```
{
  "assigned_assets": {
    "items": [
      {
        "_id": "b1c497ce.6e83759b.7pm4o13vt.t2allb0.m95ris.skmo4va1v71vfi7pqctlq"
      },
      {
        "_id": "b1c497ce.6e83759b.7pm4o130p.89ei8iu.ftq0qm.09svmjl0vtmp02dkrb4hf"
      }
    ]
  }
}
```


Remove All Assigned Assets

Verb: **PUT**
Sample URL: https://win764-vmware:9443/ibm/iis/igc-rest/v1/assets/6662c0f2.e1b1ec6c.7pm4rcajd.a36jlvd.dviqti.n8eqmjtbsbg76t59mhujr
Body:

```
{
  "assigned_assets": {
    "items": [],
    "mode": "replace"
  }
}
```


Remove All Labels

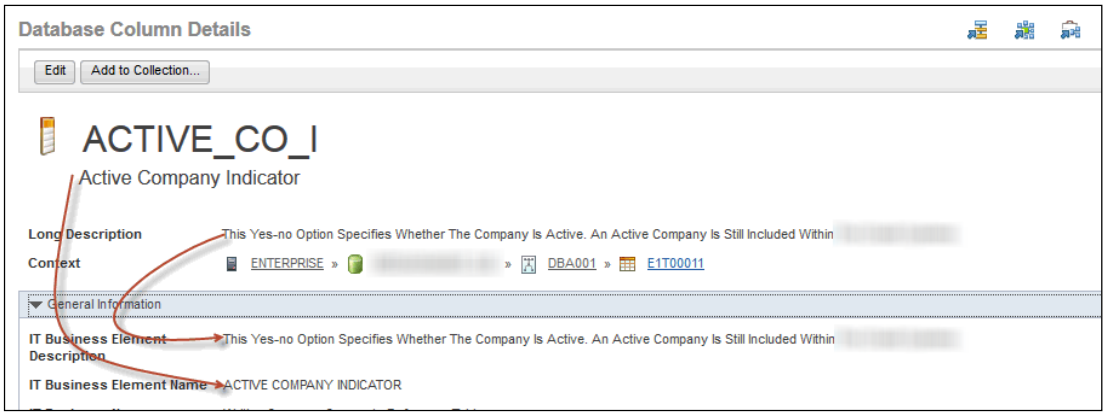
Verb: **PUT**
Sample URL: https://win764-vmware:9443/ibm/iis/igc-rest/v1/assets/6662c0f2.e1b1ec6c.7pm4rcajd.a36jlvd.dviqti.n8eqmjtbsbg76t59mhujr
Body:

```
{
  "labels": {
    "items": [],
    "mode": "replace"
  }
}
```

Use Cases

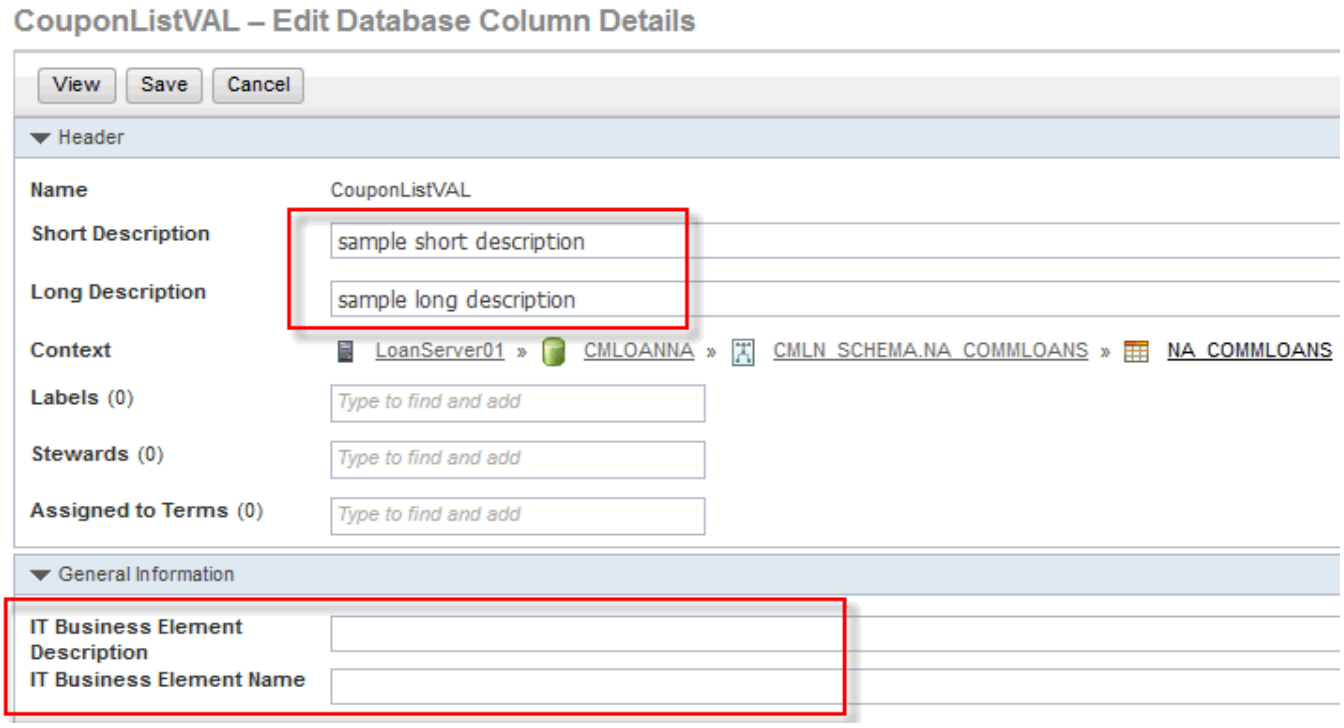
Editing asset properties

Assign a value to an information asset, such as a database table or database column property, based on another property for that same asset. In the example below, you want to set IT Business Element Name, a custom attribute, with the short description. You also want to set IT Business Element Description, a custom attribute, with the long description.



This use case uses an edit request. In this case we are copying values from one property to another. This would require two calls. First, a call using a GET, in order to obtain the values desired, and then another call with a PUT, to perform the update.

Here is a Details page, in Edit mode, of the database column asset showing the short description, long description, and the empty custom attributes:



Here is the **GET /assets/{id}** command:

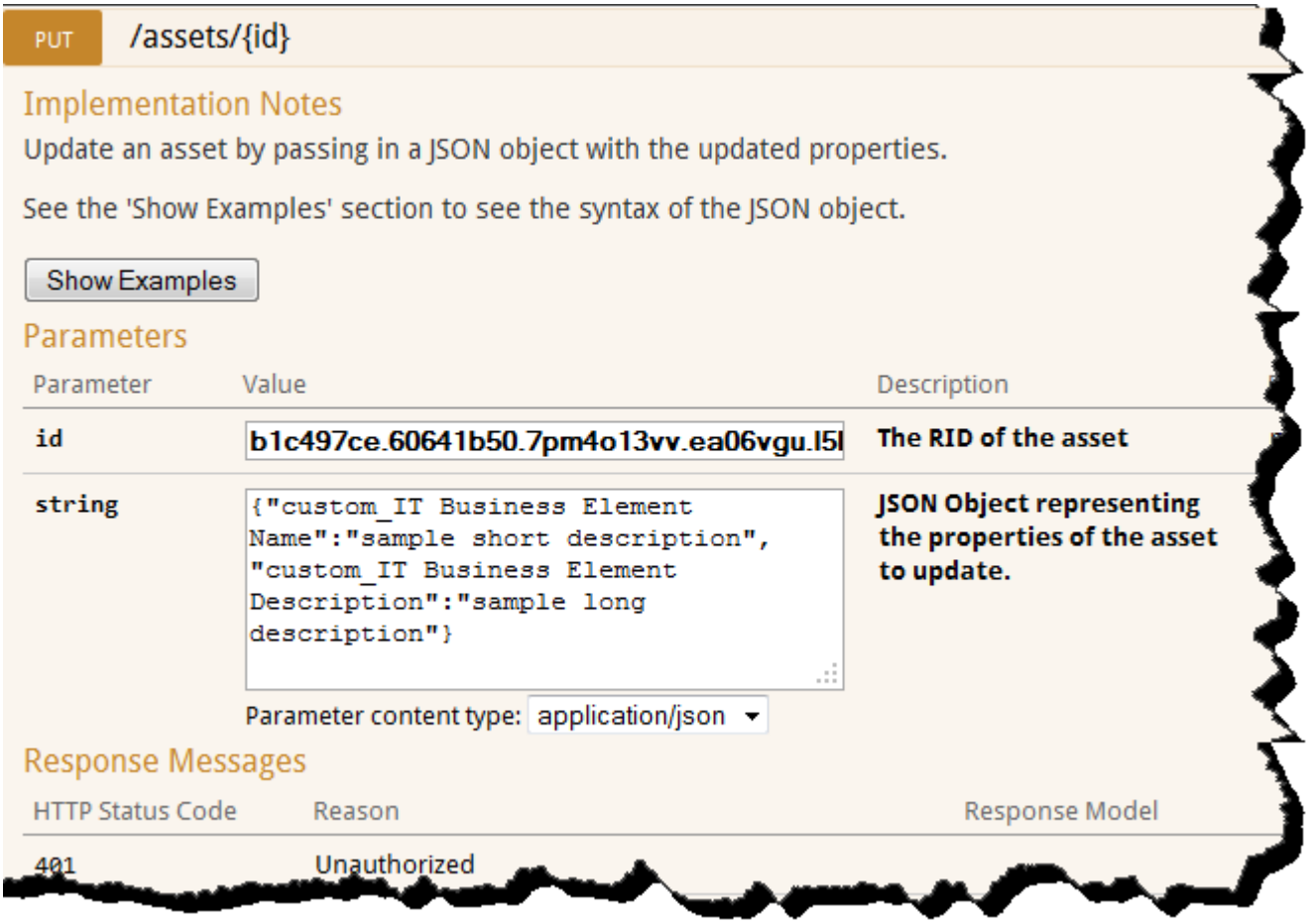
Note that the command returns a list of properties and their values for this asset. The RID in blue is the identifier for this database column. The RID can be obtained in several ways, such as from query results that were exported to a comma-separated value (CSV) file in InfoSphere Information Governance Catalog. A query can include the asset RID into its results.

After the values are obtained, the **PUT/assets/{id}** command does the update.

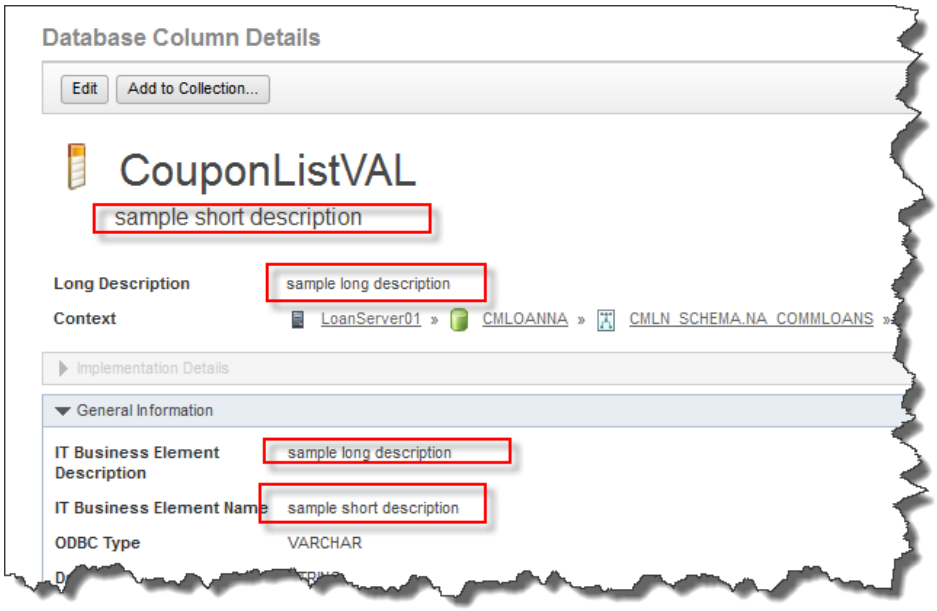
Verb: PUT
Sample URL: https://win764-vmware:9443/ibm/iis/igc-rest/v1/assets/
b1c497ce.60641b50.7pm4o13vv.ea06vgu.l5kdj0.f2tfnn7v7enart4ati7cn

Body: (which is constructed dynamically after issuing the first GET)**
{
"custom_IT Business Element Name":"sample short description", prepend custom_ to the custom attribute name
"custom_IT Business Element Description":"sample long description"
}

** The names for the Custom Attributes are found by using the **GET/types/** calls in InfoSphere Information Governance Catalog REST API.



In InfoSphere Information Governance Catalog, the Details page of the database column shows the new values for IT Business Element Name and for IT Business Element Description:

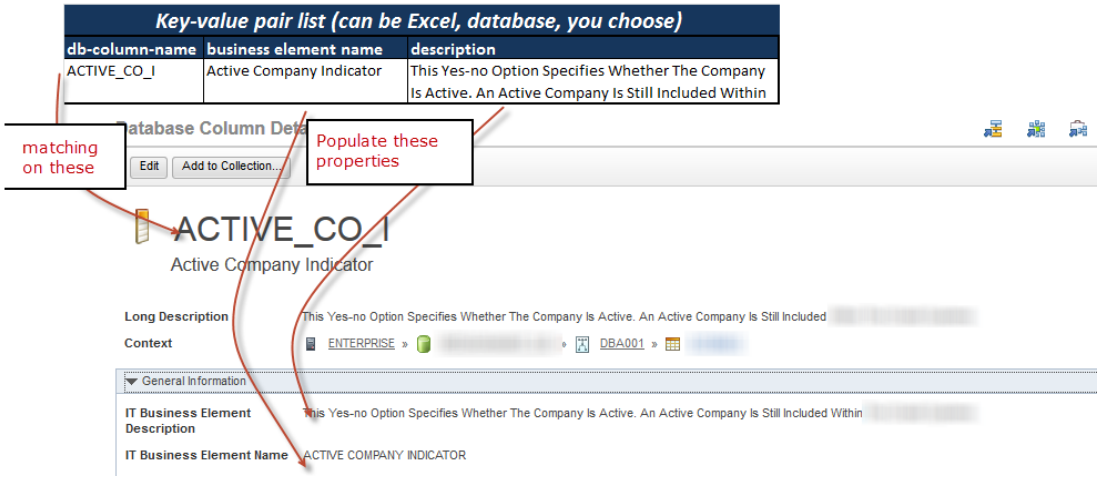


Note: If the edit is a single invocation, then an ETL tool like InfoSphere DataStage might be a good choice for performing these calls. However, if the activity requires multiple calls in succession, you might be better off writing java or other tooling to invoke REST web services, where you can more easily make multiple calls with the same connection, perform them in a loop, etc.

Assign a Value to an Information Asset Property

We want to assign a value to a property that is based on matching the name of that asset to a key-value pair. We have a key-value pair list of standard data elements, such as business element name, db-column-name, and description. This key-value pair list can be maintained in a spreadsheet file, a database table, or any another method. Matching the database column name to the db-column-name of that key-value pair list populates the properties of the database column (IT Business Element Name with the business element name from the list, and IT Business Element Description with the description from the key-value pair list).

See the following figure of the Details page of the database column asset:



The approach for this use case is effectively the same for Batch Edits, except that the initial **GET/assets/{id}** command is not needed.

This update procedure could be driven from a "ColumnName, RID" .CVS file, produced from a query. The ColumnName in the exported query results file would drive the lookup into your "key-value pair" file to obtain the correct strings for update. Then, you issue the **PUT/assets/{id}/{property}** command, as outlined below, for the various properties and/or custom attributes that you want to update.

Note on performing batch updates with InfoSphere Information Governance Catalog REST API
An ETL tool like InfoSphere DataStage is effective for applying a single InfoSphere Information Governance Catalog REST API update, as in Assign a value to an Information Asset property.

If you need to make multiple related REST API calls in succession, it might be simpler to use a language such as Java. The ETL code that is required in the flow or in the Assembly Editor of the Hierarchical Data stage of InfoSphere DataStage will get very cumbersome.

Retrieve Details of a Term from the Development Log

We want to get the information that is found in a term's Details page by using REST API rather than by using InfoSphere Information Governance Catalog.


```
{
  "development_log": {
    "items": [
      {
        "person": "Ortal Nizri",
        "new_state": "Draft",
        "workflow_task": "EDIT",
        "comment": "comment long desc",
        "date": "2015-08-24T15:45:17Z",
        "activity": "EDIT",
        "changes": [
          {
            "new_value": "a",
            "old_value": "b",
            "property": "Long Description"
          }
        ]
      }
    ],
    "paging": {
      "numTotal": 1,
      "beginIndex": 0,
      "endIndex": 1,
      "pageSize": 1
    }
  }
}
```

Retrieve Term History from the Development Log

We want to get the term history by using REST API rather than by using InfoSphere Information Governance Catalog.


```
{
  "history": {
    "items": [
      {
        "editedBy": "Ortal Nizri",
        "date": "2015-09-10T11:47:39Z",
        "comment": "will check this",
        "changes": [
          {
            "new_value": "a",
            "old_value": "b",
            "property": "Short Description"
          },
        ]
      },
      {
        "editedBy": "Ortal Nizri",
        "date": "2015-08-26T11:34:31Z",
        "comment": "A term was edited when the glossary was published"
      },
      {
        "editedBy": "Ortal Nizri",
        "date": "2015-08-23T17:42:09Z",
        "comment": "A term property or relationship was edited"
      }
    ],
    "paging": {
      "numTotal": 3,
      "beginIndex": 0,
      "endIndex": 3,
      "pageSize": 3
    }
  }
}
```

Complex Queries

Search with POST

Search for all Terms whose name **starts with 'A'**, are labeled with a label set to 'measure', are assigned to a term **that ends in 'Account'**, have custom attribute value of high for security level, contain the word 'description' in the short description, **have a modified_on date that is between two values (entered in UNIX time)**, **have a rid that can be found in a list of rids**, a parent_category rid that is equal to given rid, **and a custom attribute number value less than 10**. Display the name, modified on date, short description, labels, and parent_category for each term.

POST: https://localhost:9443/ibm/iis/igc-rest/v1/search HTTP/1.1
Accept: application/json
Host: localhost:9443
Accept-Encoding: identity

```
{
 "properties" : [ "name", "modified_on", "short_description", "labels", "parent_category" ],   array of return props
 "types" : [ "term", "database_table" ],                                                       array of object types to search
 "where" : {
  "conditions" : [ {                    array of "conditions", each in { }
   "property" : "labels.name",          inspect the list of labels, for one whose "name" property is "measure"
   "operator" : "like {0}",             appears to be equivalent to '='
   "value" : "measure"
  }, {
   "property" : "name",
   "operator" : "like {0}%" ,           string "A" is 'in front' of 'the rest' (%)
   "value" : "A"
  }, {
   "property" : "assigned_to_terms.name",      inspect the assigned_to_terms list, looking at their names
   "operator" : "like %{0}" ,                  string "Account" is at the end of the term name
   "value": "Account"
  }, {
   "property" : "custom_securitylevel",        inspect the securitylevel custom attribute for these objects
   "operator" : "like %{0}%",                  string "high" can appear anywhere in the custom attribute
   "value" : "high"
  }, {
   "property" : "modified_on",
   "operator" : "isNull",                      have nothing defined for "modified on"
   "negated" : true                            negated? ...makes this the same as "notNull"
  }, {
   "property" : "long_description",            have nothing defined for the long description
   "operator" : "isNull"
  }, {
   "property" : "_id",                         where "value" is an array of RIDs, Operator "in" will
   "operator" : "in",                          look through that list
   "value" : [ "6662c0f2.e1b1ec6c.p764l98h8.1ij2a5e.rbiis2.nbbuhdabo7mtpc6c6iupg" ]
  }, {
```

```
    "property" : "parent_category._id",
    "operator" : "=",
    "value" : [ "6662c0f2.ee6a64fe.p764l98h5.t2ve0u2.1dru7c.v9mnvhaaqjcd1r93fv9j0" ]
  }, {
    "property" : "short_description",
    "operator" : "containsWord",                a very interesting operator
    "value" : "description"
  }, {
    "min" : 1,
    "max" : 1420108720747,                      UNIX based date integers
    "property" : "modified_on",
    "operator" : "between"
  }, {
    "property" : "custom_number",               custom attribute called 'number'
    "operator" : "<=",
    "value" : "10"
  } ],
  "operator" : "and"                            Boolean function to be performed among the conditions
 }
}
```

Example with Nested Conditions

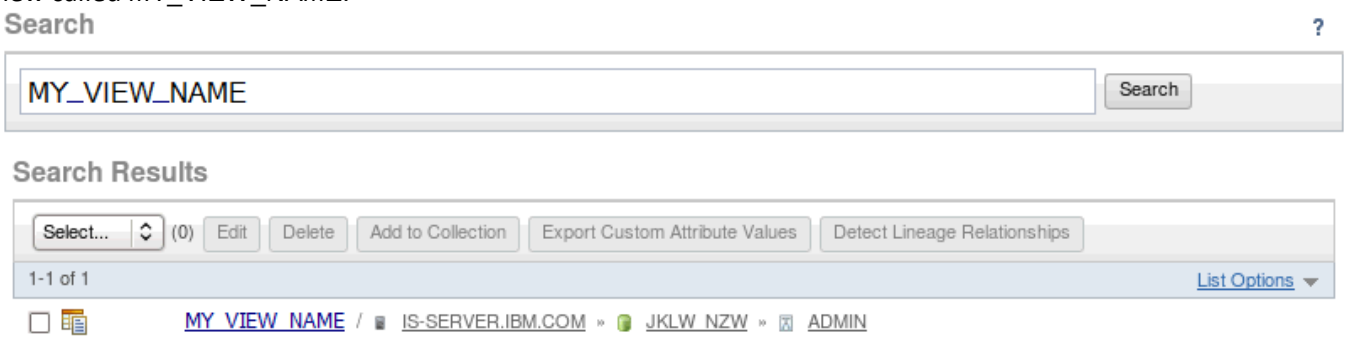
In pseudo-SQL, the below condition structure means:

where ((short_description like "%EVA%" **or** name like "EVA%") **and** custom_Golden = "YES")

```
{
 "properties": [
  "name",
  "short_description"
 ],
 "types": [
  "term"
 ],
 "where": {
  "conditions": [
   {
    "conditions": [
     {
      "property": "short_description",
      "operator": "like {%0%}",
      "value": "EVA"
     },
     {
      "property": "name",
      "operator": "like {0}%",
      "value": "EVA"
     }
    ],
    "operator": "or"
   },
   {
    "property": "custom_Golden",
    "operator": "=",
    "value": "YES"
   }
  ],
  "operator": "and"
 }
}
```

Converting a Simple Query to a REST API command

In the Search tab of InfoSphere Information Governance Catalog, you searched for, and found, a single database view called MY_VIEW_NAME.



You want to use the REST API to do the search. First, find the property type names for REST API. Then, type the following text into the **body** field of **POST/search/** :

```
{
 "properties": [
  "database_columns.name"
 ],
 "types": [
  "view"
 ],
 "where": {
  "conditions": [
   {
    "property": "name",
    "operator": "=",
    "value": "MY_VIEW_NAME"
   }
  ],
  "operator": "and"
 }
}
```