CMSCONSTRUCT

IT Discovery Machine

REST API Reference

Contents

Introduction 4

User Access 4

Request Format 4

Base Format 4

Conditional Headers 5

Conditional Payload 6

Query Filters 6

Query Comments 8

Object Identification 10

Resources 13

/itdm/data 15

/itdm/query 15

/itdm/task 16

/itdm/tool 16

/itdm/job 17

/itdm/config 18

/itdm/archive 20

Examples 22

Using a REST utility 22

GET: /data/IpAddress 27

POST: /data/IpAddress 28

GET: /query/this (IpAddress with filter) 28

PUT: /data/IpAddress/<id> 30

GET: /data/IpAddress/<id> 30

DELETE: /data/IpAddress/<id> 31

POST: /task 32

GET: /query/this (filter on description) 34

DELETE: /task 35

GET: /query/this (Flat format) 38

GET: /query/this (Nested-Simple format) 41

GET: /query/this (Nested format) 45

DELETE: /query/this 49

GET: /query/simple 51
GET: /query/simple/<name> 51
GET: /tool/count/<type> 52
GET: /tool/countExclusive/<type> 52
GET: /tool/link/<id> 53
GET: /job/runtime/<service>/filter 55
GET: /job/runtime/<service>/filter (count) 56
GET: /job/runtime/<service>/<job> 57
GET: /job/runtime/<service>/<job>/stats 58
GET: /config/Realm 59
GET: /config/NetworkScope 59
GET: /config/RealmScope 61
GET: /config/ConfigDefault 62
GET: /config/ConfigDefault/custom 62
GET: /config/ConfigGroups 63
GET: /config/OsParameters 63
GET: /config/ApiConsumerAccess 63
GET: /config/ApiConsumerAccess/testHarness 64
GET: /config/client 64
GET: /config/client/<name> 65
GET: /config/search 65
GET: /config/search/ContentGatheringResults 66

Introduction

This document provides a reference to the RESTful API for IT Discovery Machine.

User Access

REST interaction requires a registered API user. To register a new API user, either add the new user through the Admin Console, or execute the following command line on the ITDM server:

```
python <installPath>/framework/lib/createApiUserUtil.py
```

This utility will generate a 32-bit hex key for the provided user.

Keep in mind that there are varying levels of access:

- GET requests to resources (excluding /config and /archive) are available to all users
- DELETE requests require delete access.
- POST and PUT requests require write access.
- And any access to /config and /archive resources, additionally require admin access.

Request Format

Base Format

The URL endpoint depends on the settings defined in your ITDM instance, as configured here:

```
<installPath>/conf/globalSettings.json
```

Let's assume your global settings lists the API server as `itdm.cmsconstruct.com`, the service port as `52707`, the transport type as `https`, and the context root as `itdm`. That would make your base URL the following:

```
https://itdm.cmsconstruct.com:52707/itdm
```

User access is provided by two headers: apiUser and apiKey. The values come from the command line utility mentioned in the User Access section above. If we used "`insomnia`" as the user and the utility provided a key of "`f6df9d52eb3f4c29947a8718145f13ca`", then the user access would be:

```
"apiUser": "insomnia"
"apiKey" : "f6df9d52eb3f4c29947a8718145f13ca"
```

Optional headers are discussed in the section below, but the base header includes the content type and the user access:

```
"Content-Type": "application/json",
"ApiUser": "insomnia",
"ApiKey" : "f6df9d52eb3f4c29947a8718145f13ca"
```

The methods available depends on the resource, but the API supports the following:

```
GET, PUT, POST, DELETE
```

Putting it all together, API interaction will use this base format:

URL:	`http://itdm.cmsconstruct.com:52707/itdm/...`
Method:	`GET/PUT/POST/DELETE`

Header:	`{ "Content-Type": "application/json"`
	`"ApiUser": "insomnia",`
	`"ApiKey" : "f6df9d52eb3f4c29947a8718145f13ca" }`

Payload:	`{ <only provided on PUT or POST operations> }`

The REST resource determines available methods as well as whether you want to add additional headers and/or payload. The following sections go into detail on the options.

Conditional Headers

This section lists conditional headers. We use camel case for the names, but they are case insensitive. All values are quoted strings, so if the data type shows Boolean or Integer, the actual value is enclosed in quotes. This is due to type enforcement on the standard Python requests library.

RemoveEmptyAttributes	
Description:	Should the result drop (i.e. filter out) empty attributes?
Data type:	Boolean
Default value:	True
Accepted values:	True, False

RemovePrivateAttributes	
Description:	Should the result drop internal/private attributes?
Data type:	Boolean
Default value:	True
Accepted values:	True, False

ResultsFormat	
Description:	Returned dataset format from a requested query
Data type:	String
Default value:	Flat
Accepted values:	Flat, Nested, Nested-Simple

ContentDeliveryMethod	
Description:	Should a query result be returned, or cached for chunking?

Data type:	String
Default value:	(result returned in full)
Values to force caching instead:	chunk, chunks, chunked, chunking, page, pages, paged, paging, pagination

ContentDeliverySize

Description:	When ContentDeliveryMethod is caching the results for chunking/paging, this value is the size of chunk in KB
Data type:	Integer
Default value:	8192 (8 MB)
Accepted Values:	any integer value

Conditional Payload

Since the payload depends on the resource and method, this section references different examples.

To directly insert a stand-alone object, send a POST request to the "/data/<object>" resource path with a data section in the content, containing the attributes. See POST: /data/IpAddress for an example.

Alternatively, you can send a POST request to the "/task" resource path, with a content section containing the data to insert. See POST: /task for an example.

To retrieve a set of results based on a query you provide, send a GET request to the "/query/this" resource path, with the query in the payload section. There are multiple instances of this in the examples section, including GET: /query/this (Flat format).

If you send a POST request to the "/config/NetworkScope" resource path, the payload will include these sections as well: realm, active, description, data. There are several examples of Network and Realm manipulation in the Administration Guide, which we won't duplicate here.

Query Filters

This section discusses filters for queries. Each object in a query can have a filter section.

A filter can be a single condition:

```
"filter": [{
        "condition": {
            "attribute": "address",
            "operator": "==",
            "value": "192.168.1.2"
        }
}]
```

The following are valid operators for a filter condition:

==	!=
<	>
<=	>=
equal	iequal
regex	iregex
isnull	betweendate

lastnumhours	lastnumminutes

Filters can be a complex expression, with levels of expressions or conditions underneath.

An expression has a couple qualifiers for the directly included sub-expressions or conditions: `operator` and `negation`. The operator is a standard "and" or "or", which defaults to "and". The negation value is either true or false (as either a json Boolean value or a string "True" or "False"), which defaults to false.

Here's a sample complex expression:

```
"filter": [{
        "operator": "or",
        "negation": false,
        "expression": [
                {
                        "condition": {
                                "attribute": "address",
                                "operator": "==",
                                "value": "192.168.1.2"
                        }
                },
                {
                        "condition": {
                                "attribute": "address",
                                "operator": "==",
                                "value": "192.168.1.3"
                        }
                }
        ]
}],
```

Filters can also be a list of conditions. A list of conditions is implicitly wrapped by an "and" operator, which means the following is really an expression with "and" between the conditions:

```
"filter": [
    {
        "condition": {
            "attribute": "address",
            "operator": "==",
            "value": "192.168.1.2"
        }
    },
    {
        "condition": {
            "attribute": "address",
            "operator": "==",
            "value": "192.168.1.3"
        }
    }
]
```

The following filter is a single negated condition:

```
"filter": [
    {
```

```
        "negation": true,
        "condition": {
            "attribute": "path_from_filesystem",
            "operator": "isnull"
        }
    }
]
```

The above technique for checking for a not-null condition, uses the "isnull" operator with negation. A simpler way to do the same, is by using the "!=" operator with the value of null:

```
"filter": [
    {
        "condition": {
            "attribute": "path_from_filesystem",
            "operator": "!=",
            "value": null
        }
    }
]
```

While this works on a not-null condition, it does not work with checking for null. Specifically, a "==" check on a value of null will not always work (due to type casting); you need to use the "isnull" operator.

The "betweendate" operator takes a value of a list of two entries: [<date1>, <date2>]. And each value is expected in the form "YYYY-MM-DD HH:mm:ss ZZZ". For example: "2018-07-26 09:39:00 UTC". The order of dates (old, new) vs (new, old) doesn't matter; either way works.

Here's an example of the between date operator:

```
"filter": [
    {
        "condition": {
            "attribute": "time_created",
            "operator": "betweendate",
            "value": ["2018-07-26 09:39:00 UTC", "2018-07-28 09:39:00 UTC"]
        }
    }
]
```

Query Comments

There are a few additional points to mention for query definitions.

1. A query needs to have the special "linchpin" attribute on one (and only one) object. The object containing this attribute is the anchor for the query and is where result processing starts.

2. Each object will have a "label" for the result set, which implicitly defaults to the value of class_name. However, if you have multiple objects of the same class_name in your query, then you must explicitly define the "label" attribute for each, with a unique name to the query. You can of course define "label" attributes for all objects, which can make it easier for down stream tools to consume the result set (e.g. change "Node" to "Server").

3. Similar to the objects, each link must also have a label… and those should be explicitly defined.

4. All label names must be unique, or you will receive an empty result set.

Object Identification

You can either use required attributes or an object ID, for object identification.

To create an object initially, you need to supply all the required attributes. To view the required attributes of a class, go to the class definition in the database schema directory on the ITDM server:

<installPath>/framework/database/schema/*

Let's use the IpAddress class as an example. For IpAddress, we look at the following file:

<installPath>/framework/database/schema/networkElement.py

Searching for the IpAddress class in that file, we see the following definition:

```
__tablename__ = 'ip_address'
_constraints = ['address', 'realm']
_captionRule = {"condition": [ "address" ]}
__table_args__ = (UniqueConstraint(*_constraints, name='ip_address_constraint'), {"schema":"data"})
object_id = Column(CHAR(32), ForeignKey(NetworkElement.object_id), primary_key=True)
address = Column(String(128), nullable=False)
realm = Column(String(256), nullable=False)
address_type = Column(String(64), nullable=True)
is_ipv4 = Column(Boolean, nullable=True)
```

Alternatively, here's a view of the table from a database browser:

#	Name	Datatype	Length/Set	Allow NULL	Default
1	**object_id**	**CHAR**	**32**	☐	**No default**
2	address	VARCHAR	128	☐	No default
3	realm	VARCHAR	256	☐	No default
4	address_type	VARCHAR	64	☑	NULL
5	is_ipv4	BOOLEAN		☑	NULL

Notice the defined constraints (_constraints) are address and realm. So, in order to initially create an object of type IpAddress, you need to provide at least those two attributes. The same applies after object creation, if you are using the attributes again for identification (instead of the object_id).

After an object has been created, you may use its object_id for reference. The ID is a 32-bit hexadecimal value established on object creation. If you supply the object_id, then you do not need to provide the attributes. And in fact the constraint values will actually be ignored if the object_id was provided and was successfully matched.

Some objects require a runtime container. The container is usually represented by a related object in the dataset, but can also be represented directly by an attribute. Let's use the SoftwarePackage class as an example. The corresponding class definition is in the following file:

<installPath>/framework/database/schema/systemElement.py

Searching for the SoftwarePackage class in that file, we see the following definition:

```
__tablename__ = 'software_package'
_constraints = ['name', 'container', 'version', 'path']
_captionRule = {"condition": [ "name" ]}
__table_args__ = (UniqueConstraint(*_constraints, name = 'software_package_constraint'), {'schema'
object_id = Column(None, ForeignKey(SystemElement.object_id), primary_key=True)
name = Column(String(256), nullable=False)
version = Column(String(256), nullable=True)
identifier = Column(String(256), nullable=True)
title = Column(String(512), nullable=True)
associated_date = Column(String(256), nullable=True)
recorded_by = Column(String(256), nullable=True)
path = Column(String(512), nullable=True)
company = Column(String(256), nullable=True)
owner = Column(String(256), nullable=True)
vendor = Column(String(256), nullable=True)
container_relationship = relationship('Node', backref = backref('software_package_objects', cascade
container = Column(CHAR(32), ForeignKey(Node.object_id), nullable=False)
```

Alternatively, here's a view of the table from a database browser:

#	Name	Datatype	Length/Set	Allow NULL	Default
1	**object_id**	**CHAR**	**32**	☐	**No default**
2	name	VARCHAR	256	☐	No default
3	version	VARCHAR	256	☑	NULL
4	identifier	VARCHAR	256	☑	NULL
5	title	VARCHAR	512	☑	NULL
6	associated_d...	VARCHAR	256	☑	NULL
7	recorded_by	VARCHAR	256	☑	NULL
8	path	VARCHAR	512	☑	NULL
9	company	VARCHAR	256	☑	NULL
10	owner	VARCHAR	256	☑	NULL
11	vendor	VARCHAR	256	☑	NULL
12	container	CHAR	32	☐	No default

Notice the defined constraints (_constraints) are name, container, version and path. Three of those are string attributes (name, version, path) and must be on the object for attribute-based identification. But version and path are nullable, so while the attributes must be present – they can be set to null. The container on the last line is different, since it's a foreign key reference to a Node's object_id. That means the SoftwarePackage object requires a Node object as its runtime container.

To satisfy the runtime container requirement, either supply the 32-bit hex in attribute form on the SoftwarePackage object (which assumes the Node already exists and you know the object_id), or you include the Node and it's required attributes in the same dataset, along with a strong link that connects the node to the software package.

The first method allows operations on a SoftwarePackage, when only providing a single object in the payload. The second method means you have at least the following three in the payload: a SoftwarePackage, a Node, and a link that connects them. Any strong link will work, including newly defined links created by new modules. However, it must be a sub-type of the StrongLink object. The default strong link is "Enclosed".

See the examples section for usage.

Resources

There are currently seven (7) main resource paths available in the API:

- data – direct methods on objects in the data schema (CIs)
- query – query for objects and relationships, save queries, cache results
- task – bulk operations on objects in the data schema
- tool – utilities (e.g. class counts, retrieving related objects, etc)
- job – available jobs along with accrued runtime statistics
- config – platform configuration and related operations
- archive – historical tracking on model metadata and instances

Tree view of the available resources, excluding methods:

/<root>/data
    /<root>/data/{className}
    /<root>/data/{className}/{object_id}
/<root>/query
    /<root>/query/this
    /<root>/query/simple
    /<root>/query/simple/{name}
    /<root>/query/config
    /<root>/query/config/simple
    /<root>/query/config/simple/{name}
    /<root>/query/config/inputdriven
    /<root>/query/config/inputdriven/{name}
    /<root>/query/endpoint
    /<root>/query/endpoint/{name}
    /<root>/query/cache
    /<root>/query/cache/{queryId}
    /<root>/query/cache/{queryId}/{chunkId}
/<root>/task
/<root>/tool
    /<root>/tool/{className}
    /<root>/tool/count
    /<root>/tool/count/{className}
    /<root>/tool/countExclusive
    /<root>/tool/countExclusive/{className}
    /<root>/tool/link/{object_id}
    /<root>/tool/mergeObject
/<root>/job
    /<root>/job/config
    /<root>/job/config/{service}
    /<root>/job/config/{service}/{jobName}
    /<root>/job/review/{service}
    /<root>/job/review/{service}/filter
    /<root>/job/review/{service}/{jobName}
    /<root>/job/runtime/{service}
    /<root>/job/runtime/{service}/filter

/<root>/job/runtime/{service}/{jobName}
/<root>/job/runtime/{service}/{jobName}/stats
/<root>/job/log
/<root>/job/log/{jobName}
/<root>/job/log/{jobName}/{endpoint}
/<root>/config
/<root>/config/Realm
/<root>/config/Realm/{realmName}
/<root>/config/RealmScope
/<root>/config/NetworkScope
/<root>/config/NetworkScope/{realm}
/<root>/config/NetworkScope/{realm}/{object_id}
/<root>/config/ConfigGroups
/<root>/config/ConfigGroups/{realm}
/<root>/config/ConfigDefault
/<root>/config/ConfigDefault/{realm}
/<root>/config/OsParameters
/<root>/config/OsParameters/{realm}
/<root>/config/ApiConsumerAccess
/<root>/config/ApiConsumerAccess/{user}
/<root>/config/client
/<root>/config/client/{endpoint}
/<root>/config/cred
/<root>/config/cred/{type}
/<root>/config/search
/<root>/config/search/{class}
/<root>/archive
/<root>/archive/model
/<root>/archive/model/{object_id}
/<root>/archive/modelSnapshot
/<root>/archive/modelSnapshot/{object_id}
/<root>/archive/modelMetadataSnapshot
/<root>/archive/modelMetadataSnapshot/{object_id}
/<root>/archive/modelSnapshotTimestamp

Query down the resource path to see the resources available. This section will show the top level only.

Start by querying the base URL:

URL:	`http://itdm.cmsconstruct.com:52707/itdm`
Method:	`GET`

Header:	`{ "Content-Type": "application/json",`
	`"apiUser": "insomnia",`
	`"apiKey" : "f6df9d52eb3f4c29947a8718145f13ca" }`

Payload:	`{}`

Response:

```
{
```

```
  "IT Discovery Machine REST API Endpoints": [
    "/itdm/data",
    "/itdm/tool",
    "/itdm/job",
    "/itdm/query",
    "/itdm/config",
    "/itdm/task",
    "/itdm/archive"
  ]
}
```

/itdm/data

Methods on objects in the data schema (CIs).

See what it provides:

URL:	`http://itdm.cmsconstruct.com:52707/itdm/data`
Method:	`GET`

Header:	`{ "Content-Type": "application/json",`
	`"apiUser": "insomnia",`
	`"apiKey" : "f6df9d52eb3f4c29947a8718145f13ca" }`

Payload:	`{}`

Response:

```
{
  "Available classes:": ["Node", "NodeServer", "Linux", ... ],

  "/data/{class}": {
    "methods": {
      "GET": "Report all entries in specified resource class.",
      "POST": "Insert entry in specified resource class.",
      "DELETE": "Remove all entries in specified resource class."
    }
  },
  "/data/{class}/{id}": {
    "methods": {
      "GET": "Report entry with specified object id from the class resource.",
      "PUT": "Update entry with specified object id from the class resource.",
      "DELETE": "Remove entry with specified object id from the class
resource."
    },
  "/data/schema/{class}": {
    "methods": {
      "GET": "Provide specified class definition from the data schema."
    }
  }}
```

/itdm/query

Query objects and relationships, save queries, cache results.

See what it provides:

URL:	`http://itdm.cmsconstruct.com:52707/itdm/query`
Method:	`GET`

Header:	{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey" : "f6df9d52eb3f4c29947a8718145f13ca" }
Payload:	{}

Response:

```
{
  "/query/this": {
    "methods": {
      "GET": "Run the query definition provided in the payload, and return the
results.",
      "DELETE": "Delete all objects returned from running the ad-hoc query
definition provided in the payload."
    }
  },
  "/query/simple": {
    "methods": {
      "GET": "Show available resources."
    }
  },
  "/query/endpoint": {
    "methods": {
      "GET": "Show available resources."
    }
  },
  "/query/config": {
    "methods": {
      "GET": "Show available resources."
    }
  },
  "/query/cache": {
    "methods": {
      "GET": "Show available resources."
    }
  }
}
```

/itdm/task

Bulk operations on objects in the data schema (CIs).

As the base URL showed, we have these methods available for "task":

```
"POST": "Validate and insert the provided JSON data set.",
"DELETE": "Remove all objects in the provided JSON data set."
```

All methods under "task" are actionable and require additional content, so this is where our browsing ends. See examples for more information.

/itdm/tool

Utilities for class counts, retrieving related objects on CIs, etc.

See what it provides:

URL:	http://itdm.cmsconstruct.com:52707/itdm/tool

Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey" : "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
  "Available Resources": {
    "/tool/count": {
      "methods": {
        "GET": "Gives a count for all classes/links in the data schema. Parent
classes include counts from all relative child classes."
      }
    },
    "/tool/count/{ClassName_or_LinkName}": {
      "methods": {
        "GET": "Gives the count of instances in the given class, along with
any subtypes. Parent classes include counts from all relative child classes."
      }
    },
    "/tool/countExclusive": {
      "methods": {
        "GET": "Give the count of all classes/links in the data schema,
excluding counts from any child classes."
      }
    },
    "/tool/countExclusive/{ClassName_or_LinkName}": {
      "methods": {
        "GET": "Give the count of instances in the given class, excluding
counts from any child classes."
      }
    },
    "/tool/link/{id}": {
      "methods": {
        "GET": "Return all objects linked to the object specified by its id."
      }
    }
  }
}
```

/itdm/job

Available jobs along with accrued runtime statistics.

See what it provides:

URL:	`http://itdm.cmsconstruct.com:52707/itdm/job`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey" : "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
  "/job/config": {
```

```
    "methods": {
      "GET": "Show available config resources."
    }
  },
  "/job/runtime": {
    "methods": {
      "GET": "Show available runtime resources."
    }
  },
  "/job/review": {
    "methods": {
      "GET": "Show available review resources."
    }
  },
  "/job/log": {
    "methods": {
      "GET": "Show available directories on the server containing (or
previously containing) job execution logs."
    }
  }
}
```

/itdm/config

Platform configuration.

Note: this resource path requires that 'admin' level access has been granted to the user account.

See what it provides:

URL:	`http://itdm.cmsconstruct.com:52707/itdm/config`
Method:	`GET`

Header:	`{ "Content-Type": "application/json",`
	`"apiUser": "insomnia",`
	`"apiKey" : "f6df9d52eb3f4c29947a8718145f13ca" }`

Payload:	`{}`

Response:

```
{
  "Available Resources": {
    "/config/Realm": {
      "methods": {
        "GET": "Report all realms."
      }
    },
    "/config/Realm/{realmName}": {
      "methods": {
        "POST": "Insert a named realm.",
        "DELETE": "Delete realm by name."
      }
    },
    "/config/RealmScope": {
      "methods": {
        "GET": "Report all entries in RealmScope."
      }
    },
    "/config/NetworkScope": {
      "methods": {
        "GET": "Report all entries in NetworkScope.",
```

```
        "POST": "Validate, transform, and insert a single user-provided scope
entry."
      }
    },
    "/config/NetworkScope/{realm}": {
      "methods": {
        "GET": "Report details of NetworkScope from the named realm.",
        "DELETE": "Delete all entries for the named realm."
      }
    },
    "/config/NetworkScope/{realm}/{object_id}": {
      "methods": {
        "GET": "Report specific NetworkScope details from the named realm,
with the provided identifier.",
        "DELETE": "Delete the specific NetworkScope on the named realm, with
the provided identifier."
      }
    },
    "/config/ConfigGroups": {
      "methods": {
        "GET": "Return all realms with ConfigGroup entries.",
        "DELETE": "Delete all ConfigGroup entries.",
        "POST": "Create a ConfigGroup for the target realm."
      }
    },
    "/config/ConfigGroups/{realm}": {
      "methods": {
        "GET": "Report the ConfigGroup details from the named realm.",
        "DELETE": "Remove entry for the named realm.",
        "PUT": "Updates entry for the named realm."
      }
    },
    "/config/ConfigDefault": {
      "methods": {
        "GET": "Return all realms with ConfigDefault entries.",
        "DELETE": "Delete all ConfigDefault entries.",
        "POST": "Create a ConfigDefault for the target realm."
      }
    },
    "/config/ConfigDefault/{realm}": {
      "methods": {
        "GET": "Report the ConfigDefault from the named realm.",
        "DELETE": "Remove the ConfigDefault entry for the named realm.",
        "PUT": "Updates the ConfigDefault entry for the named realm."
      }
    },
    "/config/OsParameters": {
      "methods": {
        "GET": "Report available names and methods for OsParameters.",
        "DELETE": "Remove all stored OsParameters.",
        "POST": "Insert new OsParameters."
      }
    },
    "/config/OsParameters/{realm}": {
      "methods": {
        "GET": "Report the contents of OsParameters from the named realm.",
        "DELETE": "Removes entry for the named realm",
        "PUT": "Updates entry for the named realm"
      }
    },
    "/config/ApiConsumerAccess": {
      "methods": {
        "GET": "Report available names and methods for ApiConsumerAccess.",
```

```
        "DELETE": "Remove all stored ApiConsumerAccess entries.",
        "POST": "Insert new ApiConsumerAccess entry."
      }
    },
    "/config/ApiConsumerAccess/{user}": {
      "methods": {
        "GET": "Report the details of the named user.",
        "DELETE": "Removes entry that contains the provided name.",
        "PUT": "Updates entry for the provided name."
      }
    },
    "/config/client": {
      "methods": {
        "GET": "Report all server endpoints running one or more OCP service
clients."
      }
    },
    "/config/client/{endpoint}": {
      "methods": {
        "GET": "Report details on the named endpoint."
      }
    },
    "/config/cred": {
      "methods": {
        "GET": "Show available protocol credential types."
      }
    },
    "/config/cred/{type}": {
      "methods": {
        "GET": "Report entries of the provided cred type.",
        "POST": "Insert entry into the provided cred type.",
        "DELETE": "Removes entries of the provided cred type."
      }
    },
    "/config/search": {
      "methods": {
        "GET": "Show available platform schema classes that can be searched."
      }
    },
    "/config/search/{platformClassName}": {
      "methods": {
        "GET": "Query specified class in platform schema, using the provided
filter.",
        "DELETE": "Delete results from specified class in platform schema,
using the provided filter."
      }
    }
  }
}
```


/itdm/archive

Historical tracking on model metadata and instances.

Note: this resource path requires that 'admin' level access has been granted to the user account.

See what it provides:

URL:	`http://itdm.cmsconstruct.com:52707/itdm/archive`
Method:	`GET`
Header:	`{ "Content-Type": "application/json",` `"apiUser": "insomnia",`

```
                "apiKey" : "f6df9d52eb3f4c29947a8718145f13ca" }
```

Payload: `{}`

Response:

```
{
  "Resources": [
    "model",
    "modelSnapshot",
    "modelMetadataSnapshot"
  ],
  "/archive/{resource}": {
    "methods": {
      "GET": "Report all entries from the specified resource",
      "POST": "Insert new entry in the specified resource."
    }
  },
  "/archive/{resource}/{id}": {
    "methods": {
      "GET": "Report all specified resource entries containing provided id"
    }
  }
}
```

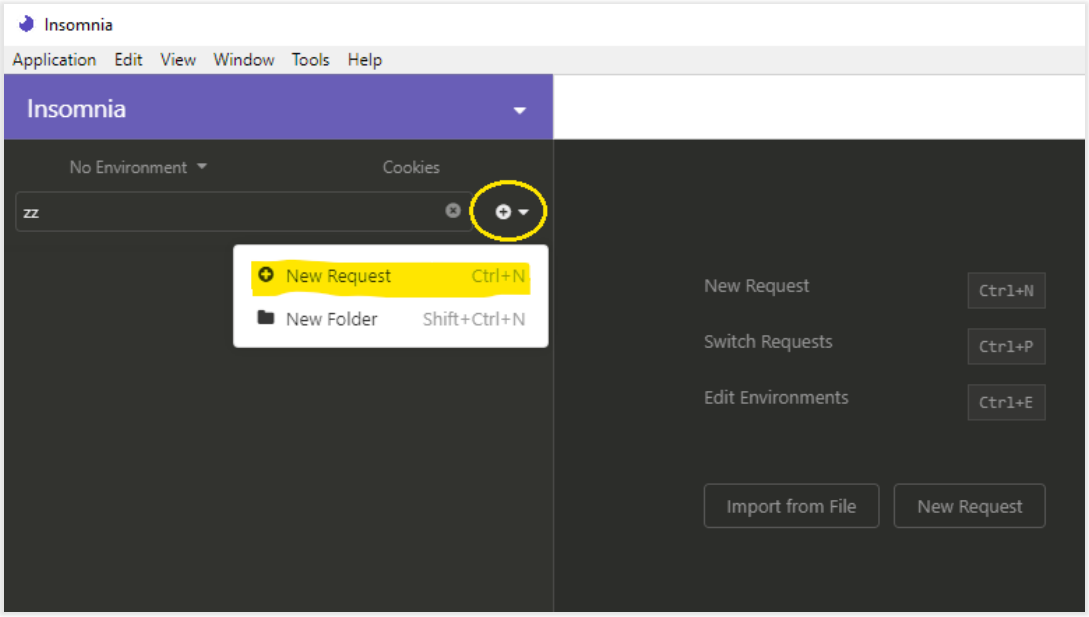
Examples

It is easier to browse web APIs with a utility instead of coding through sample queries. If you don't have a REST utility installed, there are plenty out there - including free ones like Insomnia, vRest, Katalan, Postman, etc. If you haven't used a REST utility before, we will show setup and usage through Insomnia for the first example. The remaining examples will just contain text. This allows the context to be searchable and enables copy/paste for those who wish to walk through them on their own.

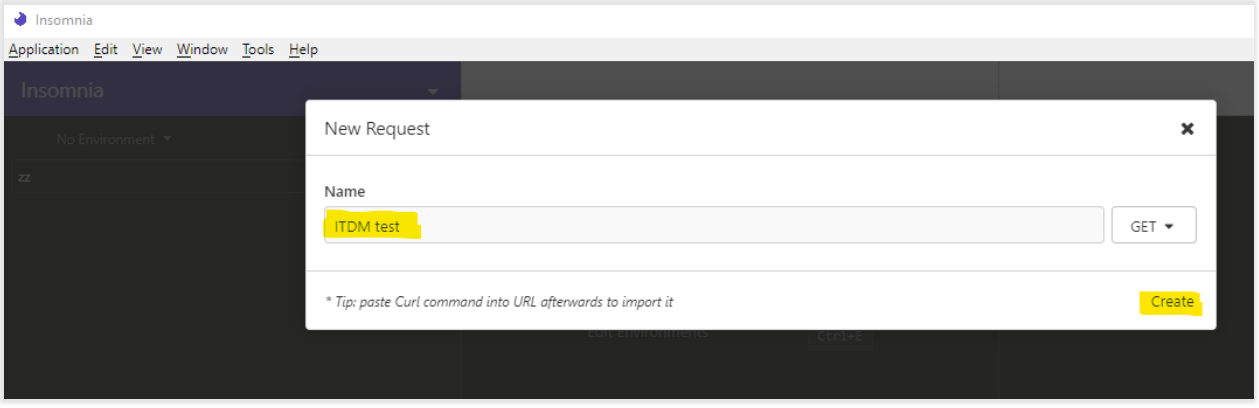
These examples provide a sampling across API resources, but make no attempt to show all methods available to a resource.

Using a REST utility

Using Insomnia for our REST utility, first we need to add a "New Request":



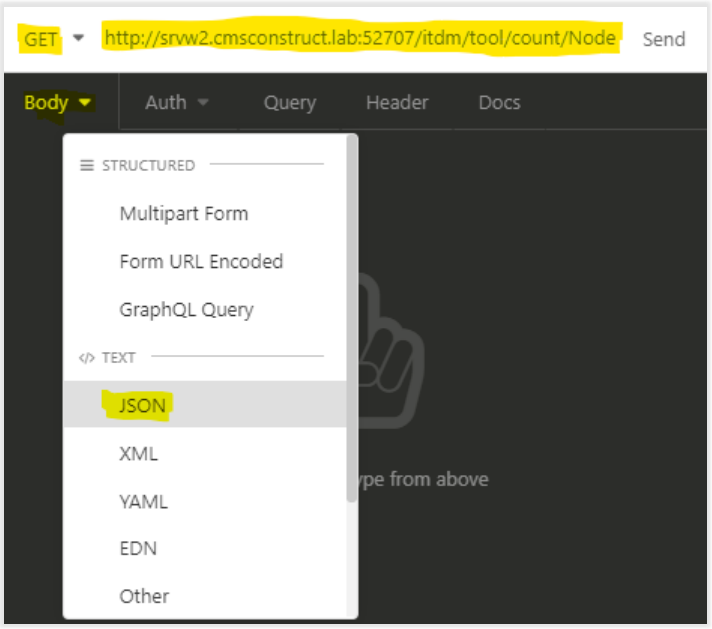
And provide a name:



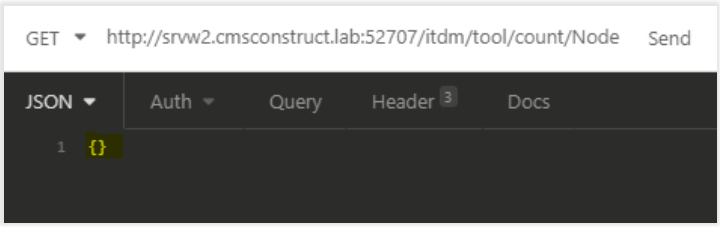
Now we configure our base connection parameters, as explained in the Request Format section above.

Verify the method is GET, and set the target URL. We'll use a lab box srvw2.cmsconstruct.lab, listening on http://srvw2.cmsconstruct.lab:52707. And we will query the /itdm/tool/count/Node resource, making our full URL: http://srvw2.cmsconstruct.lab:52707/itdm/tool/count/Node.

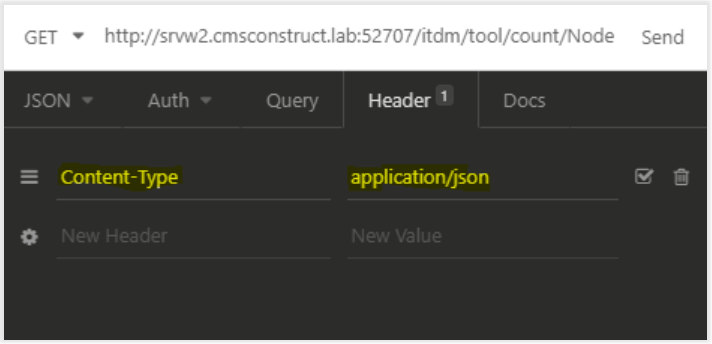
After typing in the URL, we could select the "Body" drop down to configure a "JSON" type payload.



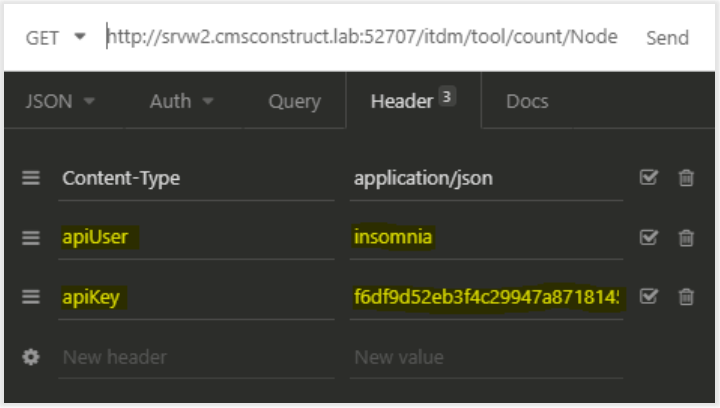
Note: We do not need to do this for our GET request, but to show how this would work (i.e. for PUT and POST requests) we can provide an empty payload:



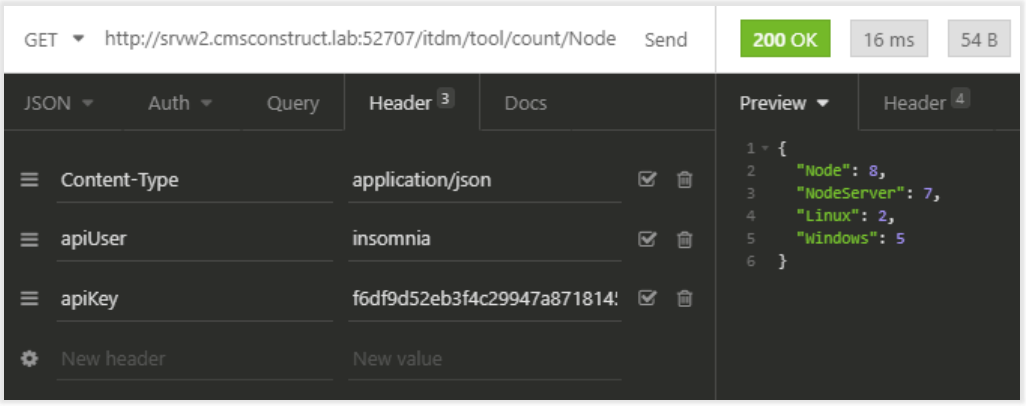
Switch tabs to the Header section and verify Content-Type is set to application/json:



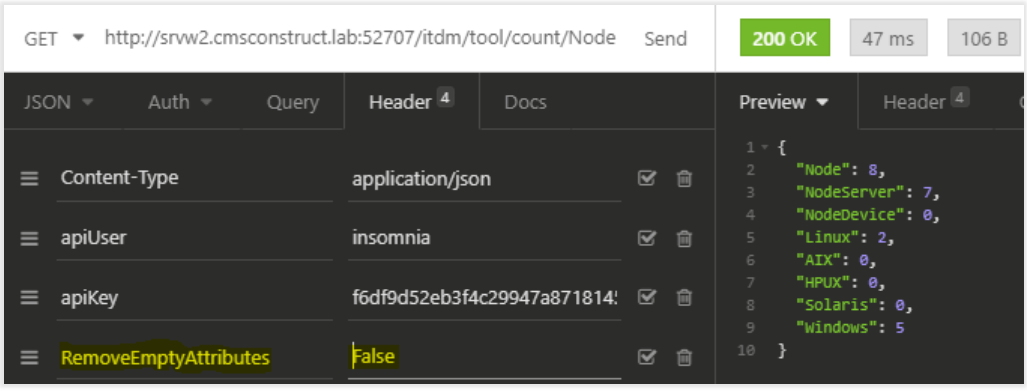
Now add two more headers, for our apiUser and apiKey:



After hitting Send, we received the following results from the lab server. Note: GET requests on the /tool/count resource path will return the count of database objects by type, where 'type' is the platform/ORM name, not the database table name. To go after a smaller result set, this example goes one level deeper for Node objects (/itdm/tool/count/Node):

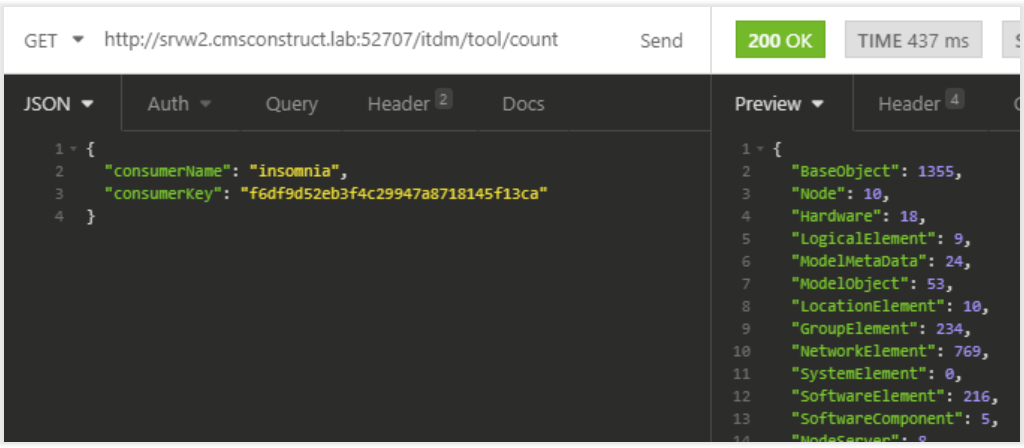


By default, the result set hides objects without any instances (i.e. Node sub-types in our case). You can override the default behavior by setting the RemoveEmptyAttributes header to False:



After doing so, we see additional Node sub-types with zero instances returned.

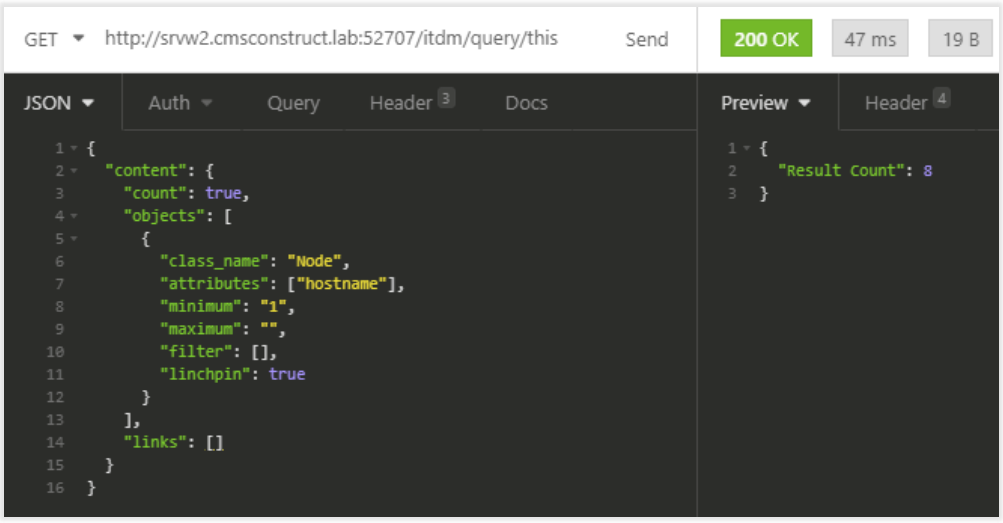
As a side note, if you set RemoveEmptyAttributes to False and query /tool/count, you will see all available object types from the data schema.



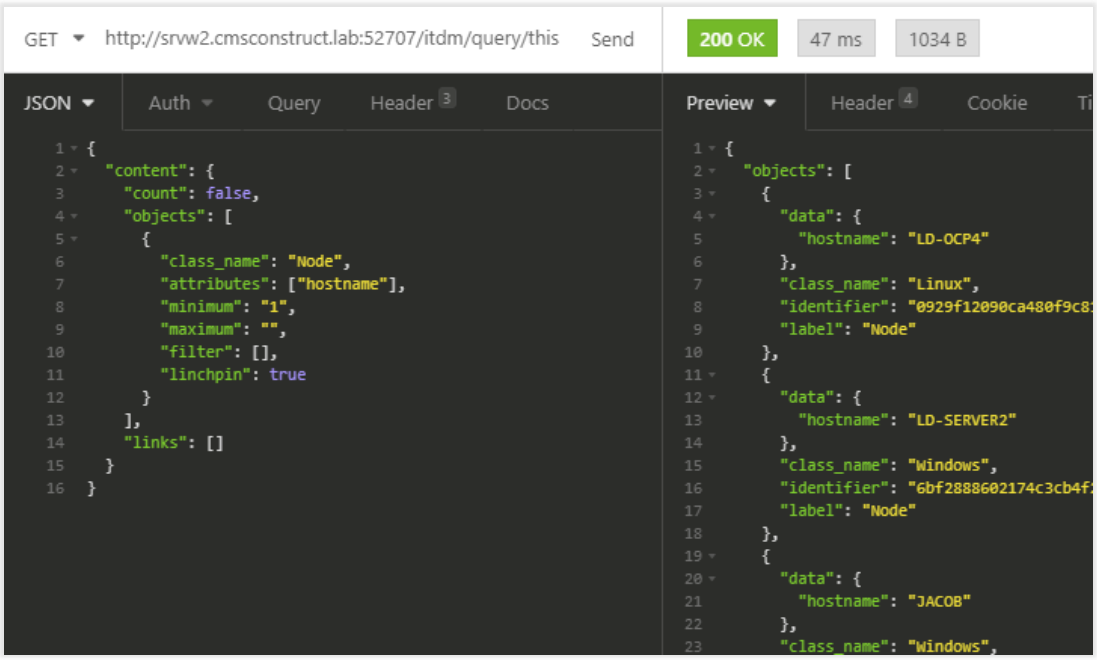
To expedite the setup of a new query, copy/paste the named query in the left pane and modify.

Quick teaser for advance queries:

We can get the same type of count from a request on the query resource. Change the resource from /tool/count/Node over to /query/this, and add a content section into the JSON, containing the desired query (objects and links). Then set 'count' as true in content, and we receive similar results:



The nice part about using this resource, is that if you change the 'count' value to false (or simply remove 'count' from the content section) - the objects are returned in the result. And the data contains only the requested attributes, instead of all attributes (returned from hitting the 'data' resource):



GET: /data/IpAddress

GET requests on a "data" resource, returns a listing of all objects for the requested resource. So by requesting the /data/IpAddress resource path, we receive all IpAddress objects:

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/data/IpAddress`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:


```
{
  "objects": [
    {
      "data": {
        "time_created": "2019-05-23 13:13:40",
        "time_updated": "2019-10-05 15:53:48",
        "time_gathered": "2019-10-05 15:51:40",
        "object_created_by": "find_ICMP_socket_test",
        "object_updated_by": "find_ICMP_socket_test",
        "description": null,
        "caption": "192.168.121.1",
        "address": "192.168.121.1",
        "realm": "default",
        "address_type": null,
        "is_ipv4": true
      },
      "class_name": "IpAddress",
      "identifier": "25f02abfcb8c4b98b9ae3e874e20bd4b",
      "label": "IpAddress"
    },
    {
      "data": {
        "time_created": "2019-05-23 13:13:40",
        "time_updated": "2019-10-21 14:57:16",
        "time_gathered": "2019-10-18 10:46:22",
        "object_created_by": "find_ICMP_socket_test",
        "object_updated_by": "dns_records",
        "description": null,
        "caption": "192.168.121.142",
        "address": "192.168.121.142",
        "realm": "default",
        "address_type": null,
        "is_ipv4": true
      },
      "class_name": "IpAddress",
      "identifier": "1c7ea1bf021c45829dc8e773e5190052",
      "label": "IpAddress"
    },
    ...
}
```

All valid attributes are returned when talking to the data resource. We can filter much of this out if we go after the query resource instead, as shown in a later example.

POST: /data/IpAddress

POST requests on a "data" resource, inserts an object of that resource type. The object is defined in the "content" section of the payload.

Request:

URL: `http://srvw2.cmsconstruct.lab:52707/itdm/data/IpAddress`

Method: `POST`

Header:
```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
```

Payload:
```
{ "content": {
      "source": "My External Provider",
      "data": {
        "address": "192.168.121.2",
        "description": "something clever",
        "is_ipv4": true,
        "realm": "default"
      }
  }
}
```

Response: Code 200 with no payload.

GET: /query/this (IpAddress with filter)

The last example created a new IpAddress object. Now we want to update that object, but first we need the object_id. We could get all IPs from /data/IpAddress again, and then look for the right one. But instead let's allow automation to do that for us, by using the /query/this resource with a filter.

GET requests on the "query/this" resource, returns the result matching a user-defined query. The query is defined in the content section of the payload.

A previous example showed a listing of IpAddress objects via the "data" resource, which returned valid attributes in the "data" section. Contrast the previous output to the following, where we show a filtered "data" section by requesting only address, realm, and description attributes in the query:

Request:

URL: `http://srvw2.cmsconstruct.lab:52707/itdm/query/this`

Method: `GET`

Header:
```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
```

Payload:
```
{ "content": {
    "objects": [{
      "class_name": "IpAddress",
      "attributes": ["address", "realm", "description"],
```

```
                "minimum": "1",
                "maximum": "",
                "linchpin": true,
                "filter": [{
                      "condition": {
                        "attribute": "description",
                        "operator": "==",
                        "value": "something clever"
                      }
                    }
                  ]
              }],
              "links": []
            }
          }
```

Response:

```
{
  "objects": [
    {
      "data": {
        "address": "192.168.121.2",
        "realm": "default",
        "description": "something clever"
      },
      "class_name": "IpAddress",
      "identifier": "cc7d032accf341c7bbe027c312d5eefb",
      "label": "IpAddress"
    }
  ]
}
```

And if you want to see all attributes, leave the attributes section empty in the above GET request (`"attributes": []`):

Response:

```
{
  "objects": [
    {
      "data": {
        "time_created": "2019-10-29 15:33:49",
        "time_updated": "2019-10-29 15:33:49",
        "object_created_by": "insomnia: My External Provider",
        "object_updated_by": "insomnia: My External Provider",
        "description": "something clever",
        "caption": "192.168.121.2",
        "address": "192.168.121.2",
        "realm": "default",
        "is_ipv4": true
      },
      "class_name": "IpAddress",
      "identifier": "cc7d032accf341c7bbe027c312d5eefb",
      "label": "IpAddress"
    }
  ]
}
```

PUT: /data/IpAddress/<id>

PUT requests on the "data" resource with a valid object type and identifier, will update the specified object. The updated context resides in the data section in the payload, as shown below.

The identifier of our IpAddress object is `"cc7d032accf341c7bbe027c312d5eefb"` from above, which makes our relative URL /data/IpAddress/cc7d032accf341c7bbe027c312d5eefb:

Request:

URL:
```
http://srvw2.cmsconstruct.lab:52707/itdm/data/IpAddress/
cc7d032accf341c7bbe027c312d5eefb
```

Method:
```
PUT
```

Header:
```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
```

Payload:
```
{ "content": {
    "source": "some other provider",
    "data": {
      "description": "wish it were something clever"
    }
  }
}
```

Response: Code 200 with no payload.

GET: /data/IpAddress/<id>

This will return details on the specified IpAddress object. We could have issued the same /query/this as shown above, but since we know the identifier, we can request it directly. Notice that the updated by and description fields changed accordingly:

Request:

URL:
```
http://srvw2.cmsconstruct.lab:52707/data/IpAddress/
cc7d032accf341c7bbe027c312d5eefb
```

Method:
```
GET
```

Header:
```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
```

Payload:
```
{}
```

Response:

```
{
  "objects": [
    {
      "data": {
        "time_created": "2019-10-29 15:33:49",
        "time_updated": "2019-10-29 15:55:49",
        "object_created_by": "insomnia: My External Provider",
        "object_updated_by": "insomnia: some other provider",
        "description": "wish it were something clever",
        "caption": "192.168.121.2",
        "address": "192.168.121.2",
```

```
          "realm": "default",
          "is_ipv4": true
        },
        "class_name": "IpAddress",
        "identifier": "cc7d032accf341c7bbe027c312d5eefb",
        "label": "IpAddress"
      }
    ]
}
```


DELETE: /data/IpAddress/<id>

To finish walking through basic object manipulation in the /data resource, we need to delete the IP. We will issue a direct delete using the object_id:

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/data/IpAddress/cc7d032accf341c7bbe027c312d5eefb`
Method:	`DELETE`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

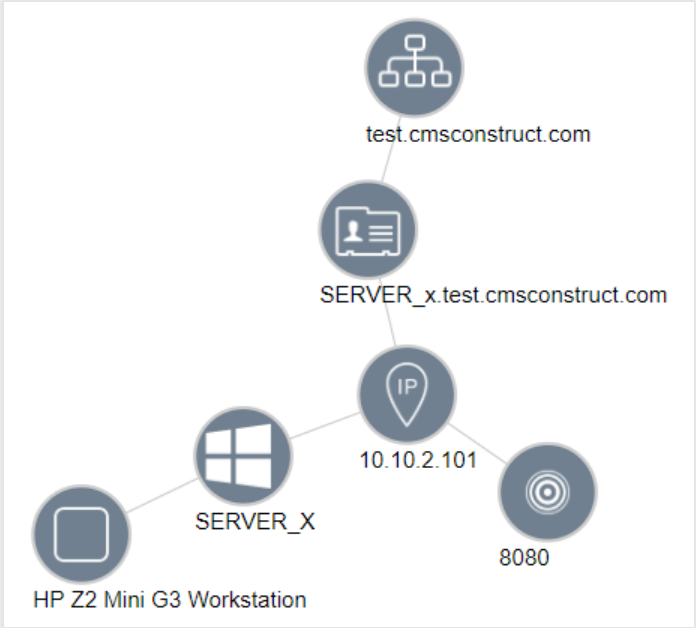
Response returns a code 200 with validation:


```
{
  "Response": "Removed entry with object_id cc7d032accf341c7bbe027c312d5eefb
from class IpAddress"
}
```

POST: /task

POST requests on the "task" resource, inserts objects defined in the map data set. Data is sent in a content section that contains CIs defined in objects and relationships defined in links. As with any PUT or POST request, we need to provide a value for source. And we will tag everything with a searchable description "Test", used to query and delete the same dataset.

Since map sets are easier to visualize than to read, here is the CI topology created by this test:



Starting top down: Domain is at the top, with a DNS record below that, followed by an IP, and the TcpPort at the bottom right. Now switching to the bottom left, we see the Hardware object connected to the Windows server, that connects to the IP.

Request:

URL: `http://srvw2.cmsconstruct.lab:52707/itdm/task`

Method: `POST`

Header:

```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
```

Payload:

```
{ "content": {
    "source": "Manual API Testing",
    "objects": [
      {
        "class_name": "Windows",
        "identifier": "1",
        "data": {
          "hostname": "SERVER_X",
          "domain": "testharness.cmsconstruct.com",
          "vendor": "Microsoft",
          "description": "Test"
        }
      },
      {
        "class_name": "IpAddress",
```

```
      "identifier": "2",
      "data": {
        "address": "10.10.2.101",
        "realm": "custom",
        "description": "Test"
      }
    },
    {
      "class_name": "HardwareNode",
      "identifier": "3",
      "data": {
        "serial_number": "2UAAAAAAAA",
        "bios_info": "N51 Ver. 01.40",
        "model": "HP Z2 Mini G3 Workstation",
        "vendor": "HPE",
        "description": "Test"
      }
    },
    {
      "class_name": "TcpIpPort",
      "identifier": "4",
      "data": {
        "name": "8080",
        "port": 8080,
        "ip": "10.10.2.101",
        "is_tcp": true,
        "description": "Test"
      }
    },
    {
      "class_name": "Domain",
      "identifier": "6",
      "data": {
        "name": "test.cmsconstruct.com",
        "description": "Test"
      }
    },
    {
      "class_name": "NameRecord",
      "identifier": "7",
      "data": {
        "name": "SERVER_x.test.cmsconstruct.com",
        "value": "10.10.2.101",
        "description": "Test"
      }
    }
  ],
  "links": [
    {
      "class_name": "Usage",
      "first_id": "1",
      "second_id": "2"
    },
    {
      "class_name": "Usage",
      "first_id": "1",
      "second_id": "3"
    },
    {
      "class_name": "Enclosed",
      "first_id": "2",
      "second_id": "4"
    },
```

```
                {
                  "class_name": "Enclosed",
                  "first_id": "6",
                  "second_id": "7"
                },
                {
                  "class_name": "Usage",
                  "first_id": "2",
                  "second_id": "7"
                }
              ]
            }
          }
```

Response: Code 200 with no payload.

GET: /query/this (filter on description)

GET requests on the "query/this" resource, returns the result matching a user-defined query. We used this resource previously with GET: /query/this (IpAddress with filter). Here we will do a high level query to the BaseObject class, filtered by "Test" in the description field:

Request:

```
URL:      http://srvw2.cmsconstruct.lab:52707/itdm/task
Method:   GET
Header:   { "Content-Type": "application/json",
            "apiUser": "insomnia",
            "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
Payload:  { "content": {
              "objects": [
                {
                  "label": "OBJECT",
                  "class_name": "BaseObject",
                  "attributes": [],
                  "filter": [{
                    "condition": {
                      "attribute": "description",
                      "operator": "==",
                      "value": "Test"
                    }
                  }],
                  "minimum": "0",
                  "maximum": "",
                  "linchpin": true
                }
              ],
              "links": []
            }
          }
```

Notice the response returns the objects, but not the links. If we wanted the connected map set, we could have constructed a more detailed query looking for connected object types. But right now we are just wanting to validate our objects went in as expected, through the /task resource.

Response:

```
{
  "objects": [
    {
      "data": {
        "caption": "10.10.2.101",
        "description": "Test"
      },
      "class_name": "IpAddress",
      "identifier": "63d7b36583414e56a9206699e2c02b09",
      "label": "OBJECT"
    },
    {
      "data": {
        "caption": "8080",
        "description": "Test"
      },
      "class_name": "TcpIpPort",
      "identifier": "4e097ba00e7247ed836d98f4d78d5705",
      "label": "OBJECT"
    },
    {
      "data": {
        "caption": "test.cmsconstruct.com",
        "description": "Test"
      },
      "class_name": "Domain",
      "identifier": "22d23e82bb724e25b323c70bc3996ea8",
      "label": "OBJECT"
    },
    {
      "data": {
        "caption": "SERVER_x.test.cmsconstruct.com",
        "description": "Test"
      },
      "class_name": "NameRecord",
      "identifier": "dc000bc315cc45ecb2df742fd5968592",
      "label": "OBJECT"
    },
    {
      "data": {
        "caption": "SERVER_X",
        "description": "Test"
      },
      "class_name": "Windows",
      "identifier": "a182e68dd33247bfb6584d3cd2c0f571",
      "label": "OBJECT"
    },
    {
      "data": {
        "caption": "HPE HP Z2 Mini G3 Workstation",
        "description": "Test"
      },
      "class_name": "HardwareNode",
      "identifier": "1899e234a6c447318603d201f14f9830",
      "label": "OBJECT"
    }
  ]
}
```


DELETE: /task

DELETE requests on the “task” resource, removes the provided objects.

Contrast this to a DELETE on the "query" type resources, where those remove objects that match a query. This method removes the specific objects provided.

To keep this simple, our content section will be the same as used for the POST. However, here are some comments on object deletion:

1. We include the links in the content section below, but that is unnecessary. Links are removed when related objects are removed.
2. We also do not need to explicitly remove objects beneath a container, if we remove the container. The TcpPort object has IpAddress as its container. Since we are removing the IpAddress, it is unnecessary to include the TcpPort; everything strongly connected to the IP will be removed when the IP is removed.

Request:

URL: `http://srvw2.cmsconstruct.lab:52707/itdm/task`

Method: `DELETE`

Header:

```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
```

Payload:

```
{ "content": {
    "objects": [
      {
        "class_name": "Windows",
        "identifier": "1",
        "data": {
          "hostname": "SERVER_X",
          "domain": "testharness.cmsconstruct.com",
          "vendor": "Microsoft",
          "description": "Test"
        }
      },
      {
        "class_name": "IpAddress",
        "identifier": "2",
        "data": {
          "address": "10.10.2.101",
          "realm": "custom",
          "description": "Test"
        }
      },
      {
        "class_name": "HardwareNode",
        "identifier": "3",
        "data": {
          "serial_number": "2UAAAAAAAA",
          "bios_info": "N51 Ver. 01.40",
          "model": "HP Z2 Mini G3 Workstation",
          "vendor": "HPE",
          "description": "Test"
        }
      },
      {
        "class_name": "TcpIpPort",
        "identifier": "4",
        "data": {
          "name": "8080",
          "port": 8080,
          "ip": "10.10.2.101",
```

```
            "is_tcp": true,
            "description": "Test"
          }
        },
        {
          "class_name": "Domain",
          "identifier": "6",
          "data": {
            "name": "test.cmsconstruct.com",
            "description": "Test"
          }
        },
        {
          "class_name": "NameRecord",
          "identifier": "7",
          "data": {
            "name": "SERVER_x.test.cmsconstruct.com",
            "value": "10.10.2.101",
            "description": "Test"
          }
        }
      ],
      "links": [
        {
          "class_name": "Usage",
          "first_id": "1",
          "second_id": "2"
        },
        {
          "class_name": "Usage",
          "first_id": "1",
          "second_id": "3"
        },
        {
          "class_name": "Enclosed",
          "first_id": "2",
          "second_id": "4"
        },
        {
          "class_name": "Enclosed",
          "first_id": "6",
          "second_id": "7"
        },
        {
          "class_name": "Usage",
          "first_id": "2",
          "second_id": "7"
        }
      ]
    }
  }
```


Response: Code 200 with no payload.

GET: /query/this (Flat format)

GET requests on the "query/this" resource, returns the result matching a user-defined query. This query is dependent on a matching data set, which is created by issuing POST: /task above.

We used the "query/this" resource previously with GET:/query/this (filter on description). Instead of just going after an object list like we did above, here we will issue a granular map set query. This will illustrate complex queries, and formatting options. In this first example we will get the results in "Flat" format, which contains a list of objects and links.

Request:

URL: `http://srvw2.cmsconstruct.lab:52707/itdm/query/this`

Method: `GET`

Header:

```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
```

Payload:

```
{ "content": {
    "objects": [{
      "label": "SERVER",
      "class_name": "Windows",
      "attributes": ["hostname", "domain", "vendor", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": "",
      "linchpin": true
    }, {
      "label": "IP",
      "class_name": "IpAddress",
      "attributes": ["address", "realm", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": ""
    }, {
      "label": "HW",
      "class_name": "HardwareNode",
      "attributes": ["serial_number", "bios_info", "model",
"vendor", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": ""
    }, {
      "label": "PORT",
```

```
      "class_name": "TcpIpPort",
      "attributes": ["name", "port", "ip", "is_tcp", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": ""
    }, {
      "label": "ZONE",
      "class_name": "Domain",
      "attributes": ["name", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": ""
    }, {
      "label": "DNS",
      "class_name": "NameRecord",
      "attributes": ["name", "value", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": ""
    }],
    "links": [{
      "label": "SERVER_TO_IP",
      "class_name": "Usage",
      "first_id": "SERVER",
      "second_id": "IP"
    }, {
      "label": "SERVER_TO_HW",
      "class_name": "Usage",
      "first_id": "SERVER",
      "second_id": "HW"
    }, {
      "label": "IP_TO_PORT",
      "class_name": "Enclosed",
      "first_id": "IP",
      "second_id": "PORT"
    }, {
      "label": "ZONE_TO_DNS",
      "class_name": "Enclosed",
      "first_id": "ZONE",
      "second_id": "DNS"
    }, {
      "label": "IP_TO_DNS",
      "class_name": "Usage",
      "first_id": "IP",
      "second_id": "DNS"
```

```
                }]
            }
        }
```

Response:


```
{
  "objects": [
    {
      "data": {
        "hostname": "SERVER_X",
        "domain": "test.cmsconstruct.com",
        "vendor": "Microsoft",
        "description": "Test"
      },
      "class_name": "Windows",
      "identifier": "897111c387b0487791838b9e8883ef56",
      "label": "SERVER"
    },
    {
      "data": {
        "address": "10.10.2.101",
        "realm": "custom",
        "description": "Test"
      },
      "class_name": "IpAddress",
      "identifier": "1d568b1a0dbe4276b05994d8e39e8747",
      "label": "IP"
    },
    {
      "data": {
        "name": "8080",
        "port": 8080,
        "ip": "10.10.2.101",
        "is_tcp": true,
        "description": "Test"
      },
      "class_name": "TcpIpPort",
      "identifier": "e6749b0cb78b455e80e79ced574dbb8c",
      "label": "PORT"
    },
    {
      "data": {
        "name": "SERVER_x.test.cmsconstruct.com",
        "value": "10.10.2.101",
        "description": "Test"
      },
      "class_name": "NameRecord",
      "identifier": "4b8274f261e947f1a89350d84f38d343",
      "label": "DNS"
    },
    {
      "data": {
        "name": "test.cmsconstruct.com",
        "description": "Test"
      },
      "class_name": "Domain",
      "identifier": "e2337db7ed9b46a28603bf2c25326846",
      "label": "ZONE"
    },
    {
      "data": {
```

```
        "serial_number": "2UAAAAAAAA",
        "bios_info": "N51 Ver. 01.40",
        "model": "HP Z2 Mini G3 Workstation",
        "vendor": "HPE",
        "description": "Test"
      },
      "class_name": "HardwareNode",
      "identifier": "47b208b282d942fcbb72b3a5ae59eff7",
      "label": "HW"
    }
  ],
  "links": [
    {
      "first_id": "897111c387b0487791838b9e8883ef56",
      "second_id": "1d568b1a0dbe4276b05994d8e39e8747",
      "class_name": "Usage",
      "label": "SERVER_TO_IP"
    },
    {
      "first_id": "1d568b1a0dbe4276b05994d8e39e8747",
      "second_id": "e6749b0cb78b455e80e79ced574dbb8c",
      "class_name": "Enclosed",
      "label": "IP_TO_PORT"
    },
    {
      "first_id": "1d568b1a0dbe4276b05994d8e39e8747",
      "second_id": "4b8274f261e947f1a89350d84f38d343",
      "class_name": "Usage",
      "label": "IP_TO_DNS"
    },
    {
      "first_id": "e2337db7ed9b46a28603bf2c25326846",
      "second_id": "4b8274f261e947f1a89350d84f38d343",
      "class_name": "Enclosed",
      "label": "ZONE_TO_DNS"
    },
    {
      "first_id": "897111c387b0487791838b9e8883ef56",
      "second_id": "47b208b282d942fcbb72b3a5ae59eff7",
      "class_name": "Usage",
      "label": "SERVER_TO_HW"
    }
  ],
  "source": "queryProcessing.py"
}
```


GET: /query/this (Nested-Simple format)

This follows the GET: /query/this (Flat format) example, but illustrates Nested-Simple result formatting. Notice the added ResultsFormat Header.

See the POST: /task example above, for the graphical view of the data set.

Nested-Simple format provides a single objects list, where the topology/hierarchy and links are seen by recursing through the children. This format allows you to recursively parse tree structures, without having to first connect the link IDs with the object IDs.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/query/this`

Method: GET

Header:
```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca",
  "ResultsFormat": "Nested-Simple" }
```

Payload:
```
{ "content": {
    "objects": [{
      "label": "SERVER",
      "class_name": "Windows",
      "attributes": ["hostname", "domain", "vendor", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": "",
      "linchpin": true
    }, {
      "label": "IP",
      "class_name": "IpAddress",
      "attributes": ["address", "realm", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": ""
    }, {
      "label": "HW",
      "class_name": "HardwareNode",
      "attributes": ["serial_number", "bios_info", "model",
"vendor", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": ""
    }, {
      "label": "PORT",
      "class_name": "TcpIpPort",
      "attributes": ["name", "port", "ip", "is_tcp", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": ""
    }, {
      "label": "ZONE",
      "class_name": "Domain",
```

```
                "attributes": ["name", "description"],
                "filter": [{
                  "condition": {
                    "attribute": "description",
                    "operator": "==",
                    "value": "Test"
                  }
                }],
                "minimum": "1",
                "maximum": ""
              }, {
                "label": "DNS",
                "class_name": "NameRecord",
                "attributes": ["name", "value", "description"],
                "filter": [{
                  "condition": {
                    "attribute": "description",
                    "operator": "==",
                    "value": "Test"
                  }
                }],
                "minimum": "1",
                "maximum": ""
              }],
              "links": [{
                "label": "SERVER_TO_IP",
                "class_name": "Usage",
                "first_id": "SERVER",
                "second_id": "IP"
              }, {
                "label": "SERVER_TO_HW",
                "class_name": "Usage",
                "first_id": "SERVER",
                "second_id": "HW"
              }, {
                "label": "IP_TO_PORT",
                "class_name": "Enclosed",
                "first_id": "IP",
                "second_id": "PORT"
              }, {
                "label": "ZONE_TO_DNS",
                "class_name": "Enclosed",
                "first_id": "ZONE",
                "second_id": "DNS"
              }, {
                "label": "IP_TO_DNS",
                "class_name": "Usage",
                "first_id": "IP",
                "second_id": "DNS"
              }]
            }
          }
```

Response:

```
[
  {
    "data": {
      "hostname": "SERVER_X",
      "domain": "test.cmsconstruct.com",
      "vendor": "Microsoft",
      "description": "Test"
```

```
  },
  "class_name": "Windows",
  "identifier": "897111c387b0487791838b9e8883ef56",
  "label": "SERVER",
  "children": [
    {
      "data": {
        "serial_number": "2UAAAAAAAA",
        "bios_info": "N51 Ver. 01.40",
        "model": "HP Z2 Mini G3 Workstation",
        "vendor": "HPE",
        "description": "Test"
      },
      "class_name": "HardwareNode",
      "identifier": "47b208b282d942fcbb72b3a5ae59eff7",
      "label": "HW",
      "link_type": "Usage",
      "link_direction": "down",
      "children": []
    },
    {
      "data": {
        "address": "10.10.2.101",
        "realm": "custom",
        "description": "Test"
      },
      "class_name": "IpAddress",
      "identifier": "1d568b1a0dbe4276b05994d8e39e8747",
      "label": "IP",
      "children": [
        {
          "data": {
            "name": "SERVER_x.test.cmsconstruct.com",
            "value": "10.10.2.101",
            "description": "Test"
          },
          "class_name": "NameRecord",
          "identifier": "4b8274f261e947f1a89350d84f38d343",
          "label": "DNS",
          "link_type": "Usage",
          "link_direction": "down",
          "children": [
            {
              "data": {
                "name": "test.cmsconstruct.com",
                "description": "Test"
              },
              "class_name": "Domain",
              "identifier": "e2337db7ed9b46a28603bf2c25326846",
              "label": "ZONE",
              "children": [],
              "link_type": "Enclosed",
              "link_direction": "up"
            }
          ]
        },
        {
          "data": {
            "name": "8080",
            "port": 8080,
            "ip": "10.10.2.101",
            "is_tcp": true,
            "description": "Test"
```

```
            },
            "class_name": "TcpIpPort",
            "identifier": "e6749b0cb78b455e80e79ced574dbb8c",
            "label": "PORT",
            "children": [],
            "link_type": "Enclosed",
            "link_direction": "down"
          }
        ],
        "link_type": "Usage",
        "link_direction": "down"
      }
    ]
  }
]
```


GET: /query/this (Nested format)

This follows the GET: /query/this (Flat format) example, but illustrates Nested result formatting. Notice the added ResultsFormat Header.

See the POST: /task example above, for the graphical view of the data set.

Nested format provides the data according to the labels. The object with the linchpin defines what will be the single list on top (as objects was the top list for the Nested-Simple format). This format is useful when you want to pinpoint objects defined in the query.

Request:

URL: `http://srvw2.cmsconstruct.lab:52707/itdm/query/this`

Method: `GET`

Header:


```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca",
  "ResultsFormat": "Nested" }
```


Payload:


```
{ "content": {
    "objects": [{
      "label": "SERVER",
      "class_name": "Windows",
      "attributes": ["hostname", "domain", "vendor", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": "",
      "linchpin": true
    }, {
      "label": "IP",
      "class_name": "IpAddress",
      "attributes": ["address", "realm", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
```

```
        }],
        "minimum": "1",
        "maximum": ""
      }, {
        "label": "HW",
        "class_name": "HardwareNode",
        "attributes": ["serial_number", "bios_info", "model",
    "vendor", "description"],
        "filter": [{
          "condition": {
            "attribute": "description",
            "operator": "==",
            "value": "Test"
          }
        }],
        "minimum": "1",
        "maximum": ""
      }, {
        "label": "PORT",
        "class_name": "TcpIpPort",
        "attributes": ["name", "port", "ip", "is_tcp", "description"],
        "filter": [{
          "condition": {
            "attribute": "description",
            "operator": "==",
            "value": "Test"
          }
        }],
        "minimum": "1",
        "maximum": ""
      }, {
        "label": "ZONE",
        "class_name": "Domain",
        "attributes": ["name", "description"],
        "filter": [{
          "condition": {
            "attribute": "description",
            "operator": "==",
            "value": "Test"
          }
        }],
        "minimum": "1",
        "maximum": ""
      }, {
        "label": "DNS",
        "class_name": "NameRecord",
        "attributes": ["name", "value", "description"],
        "filter": [{
          "condition": {
            "attribute": "description",
            "operator": "==",
            "value": "Test"
          }
        }],
        "minimum": "1",
        "maximum": ""
      }],
      "links": [{
        "label": "SERVER_TO_IP",
        "class_name": "Usage",
        "first_id": "SERVER",
        "second_id": "IP"
      }, {
```

```
                "label": "SERVER_TO_HW",
                "class_name": "Usage",
                "first_id": "SERVER",
                "second_id": "HW"
              }, {
                "label": "IP_TO_PORT",
                "class_name": "Enclosed",
                "first_id": "IP",
                "second_id": "PORT"
              }, {
                "label": "ZONE_TO_DNS",
                "class_name": "Enclosed",
                "first_id": "ZONE",
                "second_id": "DNS"
              }, {
                "label": "IP_TO_DNS",
                "class_name": "Usage",
                "first_id": "IP",
                "second_id": "DNS"
              }]
            }
          }
```

Response:

```
{
  "SERVER": [
    {
      "data": {
        "hostname": "SERVER_X",
        "domain": "test.cmsconstruct.com",
        "vendor": "Microsoft",
        "description": "Test"
      },
      "class_name": "Windows",
      "identifier": "897111c387b0487791838b9e8883ef56",
      "label": "SERVER",
      "children": {
        "IP": [
          {
            "data": {
              "address": "10.10.2.101",
              "realm": "custom",
              "description": "Test"
            },
            "class_name": "IpAddress",
            "identifier": "1d568b1a0dbe4276b05994d8e39e8747",
            "label": "IP",
            "children": {
              "PORT": [
                {
                  "data": {
                    "name": "8080",
                    "port": 8080,
                    "ip": "10.10.2.101",
                    "is_tcp": true,
                    "description": "Test"
                  },
                  "class_name": "TcpIpPort",
                  "identifier": "e6749b0cb78b455e80e79ced574dbb8c",
                  "label": "PORT",
                  "children": {},
```

```
                    "link_type": "Enclosed",
                    "link_direction": "down"
                  }
                ],
                "DNS": [
                  {
                    "data": {
                      "name": "SERVER_x.test.cmsconstruct.com",
                      "value": "10.10.2.101",
                      "description": "Test"
                    },
                    "class_name": "NameRecord",
                    "identifier": "4b8274f261e947f1a89350d84f38d343",
                    "label": "DNS",
                    "link_type": "Usage",
                    "link_direction": "down",
                    "children": {
                      "ZONE": [
                        {
                          "data": {
                            "name": "test.cmsconstruct.com",
                            "description": "Test"
                          },
                          "class_name": "Domain",
                          "identifier": "e2337db7ed9b46a28603bf2c25326846",
                          "label": "ZONE",
                          "children": {},
                          "link_type": "Enclosed",
                          "link_direction": "up"
                        }
                      ]
                    }
                  }
                ]
              },
              "link_type": "Usage",
              "link_direction": "down"
            }
          ],
          "HW": [
            {
              "data": {
                "serial_number": "2UAAAAAAAA",
                "bios_info": "N51 Ver. 01.40",
                "model": "HP Z2 Mini G3 Workstation",
                "vendor": "HPE",
                "description": "Test"
              },
              "class_name": "HardwareNode",
              "identifier": "47b208b282d942fcbb72b3a5ae59eff7",
              "label": "HW",
              "link_type": "Usage",
              "link_direction": "down",
              "children": {}
            }
          ]
        }
      }
    ]
}
```

DELETE: /query/this

DELETE requests on the "query/this" resource, removes any results matching the user-defined query. This query is dependent on a matching data set, which is created by issuing POST: /task above.

Request:

URL: `http://srvw2.cmsconstruct.lab:52707/itdm/query/this`

Method: `DELETE`

Header:

```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
```

Payload:

```
{ "content": {
    "objects": [{
      "label": "SERVER",
      "class_name": "Windows",
      "attributes": ["hostname", "domain", "vendor", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": "",
      "linchpin": true
    }, {
      "label": "IP",
      "class_name": "IpAddress",
      "attributes": ["address", "realm", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": ""
    }, {
      "label": "HW",
      "class_name": "HardwareNode",
      "attributes": ["serial_number", "bios_info", "model",
"vendor", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
          "value": "Test"
        }
      }],
      "minimum": "1",
      "maximum": ""
    }, {
      "label": "PORT",
      "class_name": "TcpIpPort",
      "attributes": ["name", "port", "ip", "is_tcp", "description"],
      "filter": [{
        "condition": {
          "attribute": "description",
          "operator": "==",
```

```
                    "value": "Test"
                  }
                }],
                "minimum": "1",
                "maximum": ""
              }, {
                "label": "ZONE",
                "class_name": "Domain",
                "attributes": ["name", "description"],
                "filter": [{
                  "condition": {
                    "attribute": "description",
                    "operator": "==",
                    "value": "Test"
                  }
                }],
                "minimum": "1",
                "maximum": ""
              }, {
                "label": "DNS",
                "class_name": "NameRecord",
                "attributes": ["name", "value", "description"],
                "filter": [{
                  "condition": {
                    "attribute": "description",
                    "operator": "==",
                    "value": "Test"
                  }
                }],
                "minimum": "1",
                "maximum": ""
              }],
              "links": [{
                "label": "SERVER_TO_IP",
                "class_name": "Usage",
                "first_id": "SERVER",
                "second_id": "IP"
              }, {
                "label": "SERVER_TO_HW",
                "class_name": "Usage",
                "first_id": "SERVER",
                "second_id": "HW"
              }, {
                "label": "IP_TO_PORT",
                "class_name": "Enclosed",
                "first_id": "IP",
                "second_id": "PORT"
              }, {
                "label": "ZONE_TO_DNS",
                "class_name": "Enclosed",
                "first_id": "ZONE",
                "second_id": "DNS"
              }, {
                "label": "IP_TO_DNS",
                "class_name": "Usage",
                "first_id": "IP",
                "second_id": "DNS"
              }]
            }
          }
```

Response: Code 200 with no payload.

GET: /query/simple

GET requests on the "query/simple" resource, provides a list of simple queries and methods available.

Instead of sending a query definition in the payload of a query request (as we do with the "query/this" resource), we can call a stored query by name. Note: queries must be stored first by sending the query definition into the appropriate resource ("query/config/simple" or "query/config/inputdriven").

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/query/simple`
Method:	`GET`

Header:	`{ "Content-Type": "application/json",`
	`"apiUser": "insomnia",`
	`"apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`

Payload:	`{}`

Response:

```
{
  "Queries": [
    "Find endpoints",
    "Models in graph form",
    "Process dependencies between nodes"
  ],
  "/query/simple/{name}": {
    "methods": {
      "GET": "Run the named query (stored previously in the database) and
return the results",
      "DELETE": "Delete all objects returned from running the named query"
    }
  }
}
```

GET: /query/simple/<name>

Get requests to a named stored resource, provides the results of the query.

Note: the default results format will be "Flat", unless you provide a valid ResultsFormat setting in the Header section (e.g. "Flat", "Nested-Simple", or "Nested).

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/query/simple/Find endpoints`
Method:	`GET`

Header:	`{ "Content-Type": "application/json",`
	`"apiUser": "insomnia",`
	`"apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`

Payload:	`{}`

Response:

```
{
```

```
  "objects": [ ... ],
  "links": [ ... ]
}
```

GET: /tool/count/<type>

Get requests to the "tool/count" resource, returns a count of the requested object type (and subtypes) from the data schema. Parent classes include counts from all relative child classes.

Given the following scenario:

- 5 base Nodes (not sub-typed)
- 25481 Linux (subtyped from Node, through NodeServer, to Linux)
- 24529 Windows (similarly subtyped Node-> NodeServer-> Windows)

Standard count returns full counts of all involved types/subtypes:

- Node: 50015
- NodeServer: 50010
- Linux: 25481
- Windows: 24529

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/tool/count/Node`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
  "Node": 12,
  "NodeServer": 10,
  "Linux": 4,
  "Windows": 6
}
```

GET: /tool/countExclusive/<type>

Get requests to the "tool/countExclusive" resource, returns a count of the requested object type from the data schema, excluding counts from any child classes.

Given the following scenario:

- 5 base Nodes (not sub-typed)
- 25481 Linux (subtyped from Node, through NodeServer, to Linux)
- 24529 Windows (similarly subtyped Node-> NodeServer-> Windows)

Exclusive count returns numbers unique to the lowest subtype:

- Node: 5,
- Linux: 25481,
- Windows: 24529

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/tool/count/Node`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
  "Node": 2,
  "Linux": 4,
  "Windows": 6
}
```

GET: /tool/link/<id>

Get requests to the "tool/countExclusive" resource, returns all objects related to the object with the specified object_id.

For this example, we'll use the identifier of the IpAddress created with a secondary run of the POST: /task example above. The IpAddress object_id for this teat was `1d568b1a0dbe4276b05994d8e39e8747`.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/tool/link/1d568b1a0dbe4276b05994d8e39e8747`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
  "source": "tool/link/1d568b1a0dbe4276b05994d8e39e8747",
  "links": [
    {
      "class_name": "Enclosed",
      "first_id": "1d568b1a0dbe4276b05994d8e39e8747",
      "second_id": "e6749b0cb78b455e80e79ced574dbb8c"
    },
    {
      "class_name": "Usage",
      "first_id": "1d568b1a0dbe4276b05994d8e39e8747",
      "second_id": "4b8274f261e947f1a89350d84f38d343"
```

```
    },
    {
      "class_name": "Usage",
      "first_id": "897111c387b0487791838b9e8883ef56",
      "second_id": "1d568b1a0dbe4276b05994d8e39e8747"
    }
  ],
  "objects": [
    {
      "class_name": "IpAddress",
      "identifier": "1d568b1a0dbe4276b05994d8e39e8747",
      "data": {
        "object_id": "1d568b1a0dbe4276b05994d8e39e8747",
        "time_created": "2019-11-01T10:16:10.777363-05:00",
        "time_updated": "2019-11-01T10:16:10.777363-05:00",
        "time_gathered": "2019-11-01 12:47:34.389018",
        "object_created_by": "insomnia: Manual API Testing",
        "description": "Test",
        "caption": "10.10.2.101",
        "address": "10.10.2.101",
        "realm": "custom",
        "address_type": null,
        "is_ipv4": null
      }
    },
    {
      "class_name": "NameRecord",
      "identifier": "4b8274f261e947f1a89350d84f38d343",
      "data": {
        "object_id": "4b8274f261e947f1a89350d84f38d343",
        "time_created": "2019-11-01T10:16:10.777363-05:00",
        "time_updated": "2019-11-01T10:16:10.777363-05:00",
        "time_gathered": "2019-11-01 12:47:34.389018",
        "object_created_by": "insomnia: Manual API Testing",
        "description": "Test",
        "caption": "SERVER_x.test.cmsconstruct.com",
        "name": "SERVER_x.test.cmsconstruct.com",
        "value": "10.10.2.101"
      }
    },
    {
      "class_name": "Windows",
      "identifier": "897111c387b0487791838b9e8883ef56",
      "data": {
        "object_id": "897111c387b0487791838b9e8883ef56",
        "time_created": "2019-11-01T10:16:10.777363-05:00",
        "time_updated": "2019-11-01T10:16:10.777363-05:00",
        "time_gathered": "2019-11-01 12:47:34.389018",
        "object_created_by": "insomnia: Manual API Testing",
        "description": "Test",
        "caption": "SERVER_X",
        "hostname": "SERVER_X",
        "domain": "test.cmsconstruct.com",
        "vendor": "Microsoft",
        "platform": null,
        "version": null,
        "hardware_is_virtual": false,
        "hardware_provider": null
      }
    },
    {
      "class_name": "TcpIpPort",
      "identifier": "e6749b0cb78b455e80e79ced574dbb8c",
```

```
      "data": {
        "object_id": "e6749b0cb78b455e80e79ced574dbb8c",
        "time_created": "2019-11-01T10:16:10.777363-05:00",
        "time_updated": "2019-11-01T10:16:10.777363-05:00",
        "time_gathered": "2019-11-01 12:47:34.389018",
        "object_created_by": "insomnia: Manual API Testing",
        "description": "Test",
        "caption": "8080",
        "name": "8080",
        "port": 8080,
        "ip": "10.10.2.101",
        "is_tcp": true,
        "port_type": null,
        "label": null
      }
    }
  ]
}
```


GET: /job/runtime/<service>/filter

Get requests to the "job/runtime/<service>/filter" resource, returns job runtime results for the service, matching the provided filter.

Notes:

- This produces similar results to calling the "config/search" resource; however, this path through the job resource does not require admin level access.
- The only difference between this request and the next one, is the `"count": true` setting in content. The next one requests just the count.

Request:

URL:
```
http://srvw2.cmsconstruct.lab:52707/itdm/job/runtime/contentGathering/filter
```

Method:
```
GET
```

Header:
```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
```

Payload:
```
{ "content": {
    "filter": [
      {
        "condition": {
          "attribute": "time_started",
          "operator": "lastnumhours",
          "value": 2
        }
      },
      {
        "condition": {
          "attribute": "job",
          "operator": "iregex",
          "value": "dynamicDiscovery"
        }
      }
    ]
  }
}
```

Response:


```
{
  "1": {
    "client_name": "SRVWBLD1",
    "consecutive_jobs_passed": 1,
    "consecutive_jobs_failed": 17,
    "date_first_invocation": "2019-09-06T16:03:02.629470",
    "date_last_failure": "2019-09-15T16:03:03.286827",
    "date_last_invocation": "2019-09-19T16:03:23.745979",
    "date_last_success": "2019-09-19T16:03:23.745979",
    "endpoint": "192.168.121.115",
    "messages": "[]",
    "result_count": {
      "Node": 1,
      "ProcessFingerprint": 76,
      "IpAddress": 4,
      "TcpIpPort": 72,
      "Enclosed": 72,
      "Usage": 73,
      "ServerClient": 1
    },
    "status": "SUCCESS",
    "time_elapsed": 10.795795,
    "time_finished": "2019-09-19T16:03:34.541774-05:00",
    "time_started": "2019-09-19T16:03:23.745979-05:00",
    "total_job_invocations": 18,
    "total_jobs_failed": 17,
    "total_jobs_passed": 1
  },

"2": {...},
"3": {...},
...
}
```


GET: /job/runtime/<service>/filter (count)

Get requests to the "job/runtime/<service>/filter" resource, returns job runtime results for the service, matching the provided filter. Issuing the same request with "count" set to true, returns just the result count.

Request:

URL: `http://srvw2.cmsconstruct.lab:52707/itdm/job/runtime/contentGathering/filter`

Method: `GET`

Header:


```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
```


Payload:


```
{ "content": {
    "count": true,
    "filter": [
      {
        "condition": {
          "attribute": "time_started",
          "operator": "lastnumhours",
          "value": 2
        }
      },
      {
```

```
                "condition": {
                  "attribute": "job",
                  "operator": "iregex",
                  "value": "dynamicDiscovery"
                }
              }
            ]
          }
        }
```

Response:

```
{
  "Result Count": 20
}
```

GET: /job/runtime/<service>/<job>

Get requests to the "job/runtime/<service>/<job>" resource, returns runtime results for the specified job.

This example uses:

- Service = contentGathering
- Job = dynamicDiscovery.powershell_app_components_voal

Note: the job name is a concatenation of the package name and the job. Doing so removes the requirement for community-developed add-on packages to have unique job names.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/job/contentGathering/dynamicDiscovery.powershell_app_components_vocal`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
  "192.168.121.115": {
    "client_name": "SRVWBLD1",
    "consecutive_jobs_passed": 1,
    "consecutive_jobs_failed": 17,
    "date_first_invocation": "2019-09-06T16:03:02.629470",
    "date_last_failure": "2019-09-15T16:03:03.286827",
    "date_last_invocation": "2019-09-19T16:03:23.745979",
    "date_last_success": "2019-09-19T16:03:23.745979",
    "endpoint": "192.168.121.115",
    "messages": "[]",
    "result_count": {
      "Node": 1,
      "ProcessFingerprint": 76,
      "IpAddress": 4,
```

```
        "TcpIpPort": 72,
        "Enclosed": 72,
        "Usage": 73,
        "ServerClient": 1
      },
      "status": "SUCCESS",
      "time_elapsed": 10.795795,
      "time_finished": "2019-09-19T16:03:34.541774-05:00",
      "time_started": "2019-09-19T16:03:23.745979-05:00",
      "total_job_invocations": 18,
      "total_jobs_failed": 17,
      "total_jobs_passed": 1
    },

  "192.168.121.140": {...},
  "192.168.121.229": {...},
  ...
  }
```


GET: /job/runtime/<service>/<job>/stats

Get requests to the "job/runtime/<service>/<job>/stats" resource, returns statistical analysis on the runtime results for the specified job.

Request:

URL:	http://srvw2.cmsconstruct.lab:52707/itdm/job/runtime/contentGathering/dynamicDiscovery.powershell_app_components_vocal/stats
Method:	GET
Header:	{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
Payload:	{}

Response:


```
{
  "Statistics": {
    "Overall": {
      "averageRuntime": 13.7165,
      "longestRuntime": 24.0162,
      "shortestRuntime": 1.3279,
      "standardDeviation": 7.6729,
      "totalEndpoints": 6
    },
    "Breakdown": {
      "FAILURE": {
          "averageRuntime": 11.8108,
          "longestRuntime": 22.2936,
          "outliers": null,
          "shortestRuntime": 1.3279,
          "standardDeviation": 10.4829,
          "totalEndpoints": 2
      },
      "SUCCESS": {
          "averageRuntime": 14.6693,
          "longestRuntime": 24.0162,
          "outliers": null,
          "shortestRuntime": 10.3179,
```

```
            "standardDeviation": 5.5354,
            "totalEndpoints": 4
        }
      }
    }
}
```


GET: /config/Realm

Get all realm names configured in the tool.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/config/Realm`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:


```
{
  "realms": {
    "default": {
      "name": "default",
      "object_id": "c51648f376d741e68dca96e8f3a04f65"
    },
    "test2": {
      "name": "custom",
      "object_id": "61156f4b05b64be7845000389559dd48"
    }
  }
}
```


GET: /config/NetworkScope

Report all entries from NetworkScope. These were provided by users (as IPs, IP ranges, or Networks) before being translated into networks included in a Realm. These are not directly used by the tool, but rather provide the bread crumb trail to determine how the RealmScope was generated.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/config/NetworkScope`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:


```
{
  "scopes": [
    {
```

```
      "active": true,
      "count": "250",
      "data": {
        "entry": "192.168.121.0/24",
        "exclusion": [
          {
            "entry": "192.168.121.14/32"
          },
          {
            "entry": "192.168.121.24"
          },
          {
            "entry": "192.168.121.34",
            "entryStop": "192.168.121.37"
          }
        ]
      },
      "description": "lab",
      "object_created_by": "insomnia: admin console",
      "object_id": 30,
      "realm": "default",
      "time_created": "2019-08-19 10:06:28",
      "transformed": "[IPv4Network('192.168.121.128/25'),
IPv4Network('192.168.121.64/26'), IPv4Network('192.168.121.48/28'),
IPv4Network('192.168.121.40/29'), IPv4Network('192.168.121.38/31'),
IPv4Network('192.168.121.32/31'), IPv4Network('192.168.121.16/29'),
IPv4Network('192.168.121.28/30'), IPv4Network('192.168.121.26/31'),
IPv4Network('192.168.121.25/32'), IPv4Network('192.168.121.0/29'),
IPv4Network('192.168.121.8/30'), IPv4Network('192.168.121.12/31'),
IPv4Network('192.168.121.15/32')]"
    },
    {
      "active": true,
      "count": "16843010",
      "data": {
        "entry": "2.2.2.2",
        "entryStop": "3.3.3.3",
        "exclusion": []
      },
      "description": "sample big net",
      "object_created_by": "insomnia: admin console",
      "object_id": 32,
      "realm": "custom",
      "time_created": "2019-08-19 10:36:44",
      "transformed": "[IPv4Network('2.2.2.2/31'), IPv4Network('2.2.2.4/30'),
IPv4Network('2.2.2.8/29'), IPv4Network('2.2.2.16/28'),
IPv4Network('2.2.2.32/27'), IPv4Network('2.2.2.64/26'),
IPv4Network('2.2.2.128/25'), IPv4Network('2.2.3.0/24'),
IPv4Network('2.2.4.0/22'), IPv4Network('2.2.8.0/21'),
IPv4Network('2.2.16.0/20'), IPv4Network('2.2.32.0/19'),
IPv4Network('2.2.64.0/18'), IPv4Network('2.2.128.0/17'),
IPv4Network('2.3.0.0/16'), IPv4Network('2.4.0.0/14'),
IPv4Network('2.8.0.0/13'), IPv4Network('2.16.0.0/12'),
IPv4Network('2.32.0.0/11'), IPv4Network('2.64.0.0/10'),
IPv4Network('2.128.0.0/9'), IPv4Network('3.0.0.0/15'),
IPv4Network('3.2.0.0/16'), IPv4Network('3.3.0.0/23'),
IPv4Network('3.3.2.0/24'), IPv4Network('3.3.3.0/30')]"
    }
  ]
}
```

GET: /config/RealmScope

Report all entries from RealmScope. These are managed and used by the tool.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/config/RealmScope`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:


```
{
  "scopes": [
    {
      "count": "250",
      "data": [
        "192.168.121.0/29",
        "192.168.121.8/30",
        "192.168.121.12/31",
        "192.168.121.15/32",
        "192.168.121.16/29",
        "192.168.121.25/32",
        "192.168.121.26/31",
        "192.168.121.28/30",
        "192.168.121.32/31",
        "192.168.121.38/31",
        "192.168.121.40/29",
        "192.168.121.48/28",
        "192.168.121.64/26",
        "192.168.121.128/25"
      ],
      "object_id": "0cb14ec5c7d14392a1171a1a76d6690d",
      "realm": "default"
    },{
      "count": "16843010",
      "data": [
        "2.2.2.2/31",
        "2.2.2.4/30",
        "2.2.2.8/29",
        "2.2.2.16/28",
        "2.2.2.32/27",
        "2.2.2.64/26",
        "2.2.2.128/25",
        "2.2.3.0/24",
        "2.2.4.0/22",
        "2.2.8.0/21",
        "2.2.16.0/20",
        "2.2.32.0/19",
        "2.2.64.0/18",
        "2.2.128.0/17",
        "2.3.0.0/16",
        "2.4.0.0/14",
        "2.8.0.0/13",
        "2.16.0.0/12",
        "2.32.0.0/11",
        "2.64.0.0/10",
        "2.128.0.0/9",
```

```
          "3.0.0.0/15",
          "3.2.0.0/16",
          "3.3.0.0/23",
          "3.3.2.0/24",
          "3.3.3.0/30"
        ],
        "object_id": "b5eb11bba8f545ee87e09c92e09905e3",
        "realm": "custom"
      }
    ]
}
```

GET: /config/ConfigDefault

List all realms with ConfigDefault entries.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/config/ConfigDefault`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
  "Realms": [ "default", "custom" ]
}
```

GET: /config/ConfigDefault/custom

Report the ConfigDefault from the named realm.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/config/ConfigDefault/custom`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
  "object_id": "5426a53b75df4ab492587f20c64045d1",
  "realm": "custom",
  "content": {
    "networkCreateLocalEndpoints": false,
    "networkResolveUnspecifiedEndpoints": true,
    "networkCreateUdpListeners": true,
    "networkCreateTcpListeners": true,
    "networkCreateTcpEstablished": true,
    "processFilterExcludeOverrideCompare": "regex",
```

```
    "processFilterExcludeOverrideList": [],
    "softwareFilterExcludeOverrideCompare": "regex",
    "softwareFilterExcludeOverrideList": []
  },
  "time_created": "2019-02-07T09:29:41.965170-06:00",
  "time_updated": "2019-08-05T10:23:36.949608-05:00",
  "object_created_by": "insomnia",
  "object_updated_by": "insomnia"
}
```

GET: /config/ConfigGroups

List all realms with ConfigGroup entries.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/config/ConfigGroups`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
  "Realms": [ "default", "custom" ]
}
```

GET: /config/OsParameters

List all realms with OsParameters entries.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/config/OsParameters`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
    "Realms": [ "default", "custom" ]
}
```

GET: /config/ApiConsumerAccess

List all account names in ApiConsumerAccess.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/config/ApiConsumerAccess`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
    "Users": [ "adminUser", "insomnia", "testHarness" ]
}
```

GET: /config/ApiConsumerAccess/testHarness

Report the details of the named account.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/config/ApiConsumerAccess/testHarness`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
  "access_admin": true,
  "access_delete": true,
  "access_read": true,
  "access_write": true,
  "active": true,
  "name": "testHarness",
  "object_created_by": "TestHarness: Test case 1.2.8.2",
  "object_id": "6fd38d9609ec4c10827f67b2518a925f",
  "object_updated_by": null,
  "owner": "ITDM Test Harness",
  "time_created": "2018-10-21 13:53:07",
  "time_updated": "2018-10-21 13:53:07"
}
```

GET: /config/client

List all previously registered service clients.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/config/client`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
  "ContentGatheringClient": [ "SRVWBLD1", "SRVWBLD2", "SRVLXBLD1" ],
  "ResultProcessingClient": [ "SRVWBLD1" ],
  "UniversalJobClient": [ "SRVWBLD1" ]
}
```

GET: /config/client/<name>

Report details on the provided service client endpoint.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/config/client/SRVWBLD1`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
  "endpointName": "SRVWBLD1",
  "platformType": "Windows-2012ServerR2-6.3.9600-SP0",
  "platformSystem": "Windows",
  "platformMachine": "AMD64",
  "platformVersion": "6.3.9600",
  "cpuType": "Intel64 Family 6 Model 62 Stepping 4, GenuineIntel",
  "cpuCount": "20",
  "memoryTotal": "24 GB",
  "health": {}
}
```

GET: /config/search

Show available platform schema classes that can be searched.

Request:

URL:	`http://srvw2.cmsconstruct.lab:52707/itdm/config/AAAAAAA`
Method:	`GET`
Header:	`{ "Content-Type": "application/json", "apiUser": "insomnia", "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }`
Payload:	`{}`

Response:

```
{
    "Available classes": [ ContentGatheringResults, UniversalJobResults, ... ]
}
```

GET: /config/search/ContentGatheringResults

Query the specified class in platform schema, using the provided filter.

Request:

URL: `http://srvw2.cmsconstruct.lab:52707/itdm/config/search/ContentGatheringResults`

Method: `GET`

Header:

```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
```

Payload:

```
{ "content": {
    "count": false,
    "filter": [
      {
        "condition": {
          "attribute": "time_started",
          "operator": "lastnumhours",
          "value": "24"
        }
      }
    ]
  }
}
```

Response:

```
{
  "1": {
    "endpoint": "192.168.121.115",
    "job": "dynamicDiscovery.powershell_app_components_vocal",
    "status": "SUCCESS",
    "messages": "[]",
    "client_name": "SRVWBLD1",
    "time_started": "2019-09-19T16:03:23.745979-05:00",
    "time_finished": "2019-09-19T16:03:34.541774-05:00",
    "time_elapsed": 10.795795,
    "result_count": {
      "Node": 1,
      "ProcessFingerprint": 76,
      "IpAddress": 4,
      "TcpIpPort": 72,
      "Enclosed": 72,
      "Usage": 73,
      "ServerClient": 1
    },
    "date_first_invocation": "2019-09-06T16:03:02.629470",
    "date_last_invocation": "2019-09-19T16:03:23.745979",
    "date_last_success": "2019-09-19T16:03:23.745979",
    "date_last_failure": "2019-09-15T16:03:03.286827",
    "consecutive_jobs_passed": 1,
    "consecutive_jobs_failed": 17,
    "total_jobs_passed": 1,
    "total_jobs_failed": 17,
    "total_job_invocations": 18
  },

"2": {...},
"3": {...},
```

```
...
}
```

Using the same query, but setting “count” in the JSON to true, shows the count instead of results:

Request:

URL: `http://srvw2.cmsconstruct.lab:52707/itdm/config/search/ContentGatheringResults`

Method: `GET`

Header:

```
{ "Content-Type": "application/json",
  "apiUser": "insomnia",
  "apiKey": "f6df9d52eb3f4c29947a8718145f13ca" }
```

Payload:

```
{ "content": {
    "count": true,
    "filter": [
      {
        "condition": {
          "attribute": "time_started",
          "operator": "lastnumhours",
          "value": "24"
        }
      }
    ]
  }
}
```

Response:

```
{
  "Result Count": 20358
}
```