11.0 Property Type APIs

WAS THIS PAGE HELPFUL? Leave Feedback

11.0 Property Type APIs

11.1 Query Property Types

Retrieve property types by query. There are two kinds of queries supported:

1. Retrieving a single property type. Retrieves exactly one property type wrapped by an XML <propertyType > element
2. Retrieving one or more property types . Retrieves 1..n property type objects wrapped by an XML <propertyTypes> collection

11.1.1 Method: GET Property Types

GET /api/propertytypes?query=(query criteria see below)

11.1.2 Method: GET a single property type by unique property type name

GET /api/propertytypes/{propertyTypeName}

11.1.3 HTTP Query and Path Parameters

Field	Type	Description	Required
query	Query	An encoded query string (where clause)	no **
first	Query	Paging. First record to start from	no
count	Query	Paging. Number of records to include when paging	no
propertyTypeName	Path	the unique name of the property type	no **

*Note: **If neither a propertyTypeName path parameter or query query parameter is not provided, all propertyTypes will be retrieved.*

11.1.4 HTTP Headers

Header	Valid Values	Required
Content-Type	application/xml or application/json	True
GWOS-API-TOKEN	a valid token returned from login	True
GWOS-APP-NAME	your application name	True

11.1.5 Query Fields

Field	Description	Alias
id	Property Type integer id	propertyTypeId
name	The property type primary unique name	
description	The description of this property type	
dataType	The data type of this property	***

Note: Query fields are case-insensitive, thus camelCase, or all lower case will both work fine.

The dataType field cannot be directly queried. It is calculated field, and must be queried with special syntax:

isBoolean = true	isInteger = true

isString = true	isLong = true
isDouble = true	

11.1.6 Example Queries

These examples are not HTTP encoded for readability. In practice queries must be encoded.

1. *query for all property types*
   GET /api/propertytypes
2. *query for all property types, order by name descending*
   GET /api/propertytypes?query=order by name desc
3. *query for a single propertyType named 'ExecutionTime'*
   GET /api/propertytypes/ExcecutionTime
4. *a like query to find all property types starting with 'RRD'*
   GET /api/propertytypes?query=name like 'RRD%'
5. *query for all property types of type INTEGER ordered by name*
   GET /api/propertytypes?query=isInteger = true order by name
6. *query for all property types of type BOOLEAN*
   GET /api/propertytypes?query=isBoolean = true
7. *query for one or more proper names using IN query syntax*
   GET /api/propertytypes?query=name in ('ExecutionTime','DeactivationTime','RRDCommand'')

Example Query Results in XML

The normal results of a query will result in either HTTP 200 OK status or a HTTP 404 NOT FOUND status.

Results of requesting a single entity with a property type name in the path parameter is always wrapped with a single `<propertyType>` entity element. Here is an XML example of the result of a query finding one property type. All fields are displayed as attributes.

```
<propertyType id="39" name="AcknowledgedBy" description="(none)"  dataType="STRING />
```

Result of queries are always wrapped in a <propertyTypes> collection element, with one or more <propertyType> subelements.

```
<propertyTypes>
  <propertyType id="39" name="AcknowledgedBy" description="(none)"  dataType="STRING" />
  <propertyType id="29" name="ExecutionTime" description="(none)"  dataType="DOUBLE" />
</propertyTypes>
```

See Appendix A for examples of usage with Curl
See Appendix B and Appendix C for example query data in both XML and JSON:
Response Data - XML - JSON

11.2 Create or Update Property Types

Persist a batch (1..n) of property types in foundation database. Property types will be created if they do not exist. If a property type exists, it will be updated.

If one or more property type creation operations fails, others may still succeed. This is not an all-or-none transactional operation. The results of each individual Property Type creation/update operation is returned back in the resultset described below with a status of success or failure.

11.2.1 Method: POST

POST /api/propertytypes

11.2.2 HTTP Headers

Header	Valid Values	Required
Content-Type	application/xml or application/json	True
GWOS-API-TOKEN	a valid token returned from login	True
GWOS-APP-NAME	your application name	True

11.2.3 Post Data Attributes and Elements

Field	Description	Required	Type
name	The unique property type name	Yes	Attribute
description	The description of this property type	No	Attribute
datatype	The data type name. Valid values are: BOOLEAN STRING INTEGER LONG DOUBLE	Yes	Attribute

11.2.4 XML POST Data Example

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<propertyTypes>
  <propertyType name="unusedCPU" description="Measures unused CPU usage" dataType="DOUBLE" />
  <propertyType name="virtualName" description="Name of virtual app" dataType="STRING" />
</propertyTypes>
```

More Post Data Examples: XML - JSON

11.2.5 HTTP Status Codes

200 - Zero or more property types were created without any internal server errors
500 - An internal server error occurred

11.2.6 Example Response

In this example, we use the POST data above. Two property types were successfully created.

Note that the results collection also returns the location of the created or updated property type which can be directly used by the GET operation.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<results successful="2" failed="0" entityType="PropertyType"
       operation="Update" warning="0" count="2">
  <result>
    <entity>unusedCPU</entity>
    <location>http://localhost/api/propertytypes/unusedCPU</location>
<status>success</status>
  </result>
  <result>
    <entity>virtualName</entity>
    <location>http://localhost/api/propertytypes/virtualName</location>
<status>success</status>
  </result>
</results>
```

11.3 Delete Property Types

Deletes one or more property types from the Collage database.

11.3.1 Method: DELETE

DELETE /api/propertytypes/name1

where *name1* is the name of the property type to delete

DELETE /api/propertytypes/name1,name2 ...
A comma-separated list of two property names (or more) are provided. Note that no-spaces are allowed in this HTTP path segment.

11.3.2 HTTP Headers

Header	Valid Values	Required
Content-Type	application/xml or application/json	True
GWOS-API-TOKEN	a valid token returned from login	True
GWOS-APP-NAME	your application name	True

11.3.3 HTTP Status Codes

200 - Property types were deleted successfully
500 - An internal server error occurred

11.3.4 Example Response

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<results successful="2" failed="0" entityType="PropertyType"
       operation="Delete" warning="0" count="2">
  <result>
    <entity>name1</entity>
    <message>Property type deleted</message>
    <status>success</status>
  </result>
  <result>
    <entity>name2</entity>
    <message>Property type deleted</message>
    <status>success</status>
  </result>
</results>
```