12.0 Application Type APIs

WAS THIS PAGE HELPFUL? Leave Feedback

12.0 Application Type APIs

12.1 Query Application Types

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

1. Retrieve a single applicationType. Retrieves exactly one applicationType wrapped XML `<applicationType>` element
2. Retrieve one or more application types . Retrieves 1..n application type objects wrapped by an XML `<applicationTypes>` collection

12.1.1 Method: GET Application Types

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

12.1.2 Method: GET a single Application Type by unique application type name

GET /api/applicationtypes/{applicationTypeName}

12.1.3 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
applicationTypeName	Path	the unique name of the application type	no **
depth	Query	The depth of the response data. Either ‘shallow’ or ‘deep’. Defaults to shallow	no

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

12.1.4 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

12.1.5 Query Fields

Field	Description	Alias
id	Application Type integer id	applicationTypeId
name	The application type primary unique name	
description	The description of this application type	
stateTransitionCriteria	Comma separated list of state transition criteria	

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

12.1.6 Example Queries

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

1. *query for all application types*
   GET /api/applicationtypes
2. *query for all application types, order by name descending*
   GET /api/applicationtypes?query=order by name desc
3. *query for a single applicationType named 'NAGIOS'*
   GET /api/applicationtypes/NAGIOS
4. *a like query to find all application types in list*
   GET /api/applicationtypes?query=name in ('SYSTEM','SNMPTRAP','SYSLOG')

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

Note that Application Types return a number of fields that are calculated and thus it is not possible to query them:

1. entityProperties
2. entityTypes
3. any capitalized property

```
<applicationType id="200" name="VEMA"
       description="Virtual Environment Monitor Agent"
       stateTransitionCriteria="Device;Host;ServiceDescription">
  <properties>
         ....
  </properties>
  <entityProperties>
      ...
  </entityProperties>
  <entityTypes>
     ...
  </entityTypes>
</applicationType>
```

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

```
<applicationTypes
  <applicationType id="107" name="SYSLOG"
       description="SYSLOG Application"
       stateTransitionCriteria="Device">
  ...
  </applicationType>
  <applicationType id="200" name="VEMA"
       description="Virtual Environment Monitor Agent"
       stateTransitionCriteria="Device;Host;ServiceDescription">
</applicationTypes>
```

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

12.2 Create or Update Application Types

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

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

Posts allow for associating entity properties with an application type to define the valid EntityType/PropertyType values for this ApplicationType.

```
<entityProperty>
  <entityType>HOST_STATUS</entityType>
  <propertyType>isAcknowledged</propertyType>
  <sortOrder>52</sortOrder>
</entityProperty>
```

12.2.1 Method: POST

POST /api/applicationtypes

12.2.2 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

12.2.3 Post Data Attributes and Elements

Field	Description	Required	Type
name	The unique application type name	Yes	Attribute
description	The description of this application type	No	Attribute
stateTransitionCriteria	comma-separated list of transition criteria	Yes	ttribute
entityProperties	A list of entityProperties defining the valid entityTypes and propertyTypes for this applicationType. An optional sort order can also be provided.	No	Attribute

12.2.4 XML POST Data Example

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<applicationTypes>
  <applicationType name="DAVE" description="Virtual Environment Monitor Agent"
       stateTransitionCriteria="Device;Host;ServiceDescription">
  <entityProperties>
    <entityProperty>
      <entityType>HOST_STATUS</entityType>
      <propertyType>isAcknowledged</propertyType>
      <sortOrder>52</sortOrder>
    </entityProperty>
    <entityProperty>
      <entityType>HOST_STATUS</entityType>
      <propertyType>LastPluginOutput</propertyType>
      <sortOrder>53</sortOrder>
    </entityProperty>
    <entityProperty>
      <entityType>SERVICE_STATUS</entityType>
      <propertyType>LastPluginOutput</propertyType>
      <sortOrder>54</sortOrder>
    </entityProperty>
    <entityProperty>
      <entityType>SERVICE_STATUS</entityType>
      <propertyType>PerformanceData</propertyType>
      <sortOrder>55</sortOrder>
    </entityProperty>
  </entityProperties>
  </applicationType>
</applicationTypes>
```

More Post Data Examples: XML - JSON

12.2.5 HTTP Status Codes

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

12.2.6 Example Response

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

Note that the results collection also returns the location of the created or updated application 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="ApplicationType"
       operation="Update" warning="0" count="2">
  <result>
    <entity>NEWAPP</entity>
    <location>http://localhost/api/applicationtypes/NEWAPP</location>
<status>success</status>
  </result>
  <result>
    <entity>NEWAPP2</entity>
    <location>http://localhost/api/applicationtypes/NEWAPP2</location>
<status>success</status>
  </result>
</results>
```

12.3 Delete Application Types

Deletes one or more application types from the Collage database.

12.3.1 Method: DELETE

DELETE /api/applicationtypes/name1

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

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

12.3.2 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

12.3.3 HTTP Status Codes

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

12.3.4 Example Response

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