3.0 Host Group APIs

WAS THIS PAGE HELPFUL? Leave Feedback

3.0 Host Group APIs

3.1 Query Host Groups

Retrieve host groups by query. There are two kinds of queries supported:

1. Retrieving a single host group. Retrieves exactly one host group object wrapped by an XML `<hostGroup>` element
2. Retrieving one or more host groups . Retrieves 1..n host group objects wrapped by an XML `<hostGroups>` collection

3.1.1 Method: GET HostGroups

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

3.1.2 Method: GET a single host group by name

GET /api/hostgroups/{hostGroupName}

3.1.3 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
hostGroupName	Path	the name of the host group	no
depth	Query	The depth of the response data. Either ‘shallow’ or ‘deep’. Defaults to shallow	no

If neither a hostGroupName path parameter or query query parameter is not provided, all hosts groups will be retrieved.

3.1.4 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

3.1.5 Query Fields

Field	Description	Alias
id	Host group identification	hostGroupId
name	The name of the host group	
description	The description of this host group	
alias	An alias for this host group	
appType	Application Type	applicationType
hosts.	A prefix for the the associated hosts with this group. For example: hosts.hostName or hosts.description	

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

3.1.6 Example Queries

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

1. *query for all host groups, with shallow depth*
   GET /api/hostgroups
2. *query for all host groups, with deep depth*
   GET /api/hostgroups?depth=deep
3. *query for a single host group named Engineering, with depth of shallow*
   GET /api/hostgroups/Engineering
4. *query for a single host group named Engineering, with depth of deep*
   GET /api/hostgroups/Engineering?depth=deep
5. *query to find all host groups containing a host named 'localhost'*
   GET /api/hostgroups?query=hosts.hostName = 'localhost'
6. *a simple like query on the host group name*
   GET /api/hostgroups?query=name like 'Eng%'
7. *query for one or more names using in query syntax*
   GET /api/hostgroups?query=name in ('Engineering','Support','IT','HG1')

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 hostGroupName in the path parameter is always wrapped with a single `<hostGroup>` entity element. Result of queries are always wrapped in a `<hostGroups>` collection element, with one or more `<hostGroup>` subelements.

Here is an XML example of the result of a query finding one host group. Properties are always wrapped in a `<properties>` collection element. All other fields are attributes.

```
<hostGroup id="5" name="Support" alias="support" appType="NAGIOS">
  <hosts>
    <host id="17" hostName="mc-cent5-64-a"
    description="mc-cent5-64-a" monitorStatus="UNSCHEDULED DOWN"
    appType="NAGIOS" deviceIdentification="172.28.113.170"
    lastCheckTime="2013-05-22T09:44:16-07:00"
    bubbleUpStatus="UNSCHEDULED DOWN" serviceAvailability="100"
    acknowledged="false" serviceCount="1" />
    <host id="6" hostName="172.28.113.151"
    description="172.28.113.151" monitorStatus="UP"
    appType="NAGIOS" deviceIdentification="172.28.113.151"
    lastCheckTime="2013-05-22T09:43:06-07:00" bubbleUpStatus="UP"
    serviceAvailability="50" acknowledged="false"
    serviceCount="4" />
    <host id="13" hostName="mc-cent5-64-5"
    description="mc-cent5-64-5" monitorStatus="UP"
    appType="NAGIOS" deviceIdentification="172.28.113.161"
    lastCheckTime="2013-05-22T09:45:26-07:00" bubbleUpStatus="UP"
    serviceAvailability="0" acknowledged="false"
    serviceCount="4" />
    <host id="12" hostName="172.28.113.168"
    description="172.28.113.168" monitorStatus="UP"
    appType="NAGIOS" deviceIdentification="172.28.113.168"
    lastCheckTime="2013-05-22T09:43:16-07:00" bubbleUpStatus="UP"
    serviceAvailability="0" acknowledged="false"
    serviceCount="3" />
  </hosts>
</hostGroup>
```

For queries, the hostGroup elements are wrappered in a hostGroups element.

```
<hostGroups>
  <hostGroup id="5" name="Support" alias="support" appType="NAGIOS">
    ...
  </hostGroup>
  <hostGroup id="6" name="Sales" alias="sales" appType="NAGIOS">
    ...
  </hostGroup>
  ...
</hostGroups>
```

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

3.2 Create Host Groups

Creates a batch (1..n) of new host groups in foundation database. Host Groups will be created if they do not exist. If a host group exists, it will be updated. You can also specify one or more hosts to be created into the host group. Note these hosts must already exist in the system. They will not be automatically created.

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

3.2.1 Method: POST

POST /api/hostgroups

3.2.2 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

3.2.3 Post Data Attributes

Field	Description	Required
name	The name of the host group	yes
description	The description of this host group	no
alias	An alias for this host group	no
appType	Application Type	no, defaults to NAGIOS
hosts	The collection of one or more hosts to add to this host group. Note these hosts must have been previously added to the system	no

3.2.4 XML POST Data Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<hostGroups>
  <hostGroup name="blueGroup" description="My Blues Group" alias="blue">
    <hosts>
      <host hostName="localhost" />
      <host hostName="notfound-host"/>
    </hosts>
  </hostGroup>
  <hostGroup name="redGroup" description="My Red Group" alias="red">
    <hosts>
      <host hostName="demo"/>
      <host hostName="localhost"/>
    </hosts>
  </hostGroup>
</hostGroups>
```

More Post Data Examples: XML - JSON

3.2.5 HTTP Status Codes

200 - Zero or more host groups were created without any internal server errors

500 - An internal server error occurred

3.2.6 Example Response

In this example, we use the POST above. Two host group objects were attempted to be created. Additionally, we tried to add two hosts to each host group. One of those hosts "notfound-host" failed to be added to the host group since it didn't exist in the system. This condition was considered a 'warning' condition.

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

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<results successful="2" failed="0" entityType="HostGroup" operation="Update" warning="1" count="3">
  <result>
    <entity>blueGroup</entity>
    <location>
    [http://localhost/api/hostgroups/blueGroup]</location>
<status>success</status>
  </result>
  <result>
    <entity>redGroup</entity>
    <location>
    [http://localhost/api/hostgroups/redGroup]</location>
<status>success</status>
  </result>
  <result>
    <entity>notfound-host</entity>
    <message>Hosts did not exist and were not processed</message>
    <status>warning</status>
  </result>
</results>
```

3.3 Delete Host Group

Deletes one or more host groups from the Collage database. There are two methods of deletion:

1. Delete without post data, passing in one or more host group names on URL path
2. Delete with post data, passing in host group names in post data

3.3.1 Method: DELETE without POST Data

DELETE /api/hostgroups/hostGroupName

where *hostName* is a single host group name to be deleted

DELETE /api/hostgroups/hostGroupName1,hostGroupName2
where *hostGroupName1*,*hostGroupName2* is a comma-separated list of host group names. Note that no-spaces are allowed in this HTTP path segment.

3.3.2 Method: DELETE with POST Data

DELETE /api/hostgroups

The post data should be formatted like a POST payload, but the only required field is the host Group name (name). Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<hostGroups>
  <hostGroup name="hostGroup-100" />
  <hostGroup name="hostGroup-101" />
</hostGroups>
```

3.3.3 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

3.3.4 HTTP Status Codes

200 - Hosts were deleted successfully
500 - An internal server error occurred

3.3.5 Example Response

```
<results count='2' success='2' failure='0' entityType='HostGroup' operation='Delete'>
  <result status='success' entity='hostGroup-100'
    location='http://localhost/monitor/api/hostgroups/hostGroup-100' />
<result status='success' entity='hostGroup-101'
    location='http://localhost/monitor/api/hostgroups/hostGroup-101' />
</results>
```

3.4 Clear Host Group

Clears the collection of hosts for one or more host groups from the Collage database. The Clear web service does not actually delete the host group. It simply deletes the associations between a host group and all of its hosts. In effect, 'clearing' the host group to an empty set of hosts. Note that the Clear operation uses the DELETE HTTP method and one query parameter named 'clear'.

There are two methods of clearing:

1. Clear without post data, passing in one or more host group names on URL path
2. Clear with post data, passing in host group names in post data

3.4.1 Method: CLEAR without POST Data

DELETE /api/hostgroups/hostGroupName?clear=true

where *hostName* is a single host group name to be cleared

DELETE /api/hostgroups/hostGroupName1,hostGroupName2?clear=true

where *hostGroupName1*,*hostGroupName2* is a comma-separated list of host group names to be cleared. Note that no-spaces are allowed in this HTTP path segment.

3.4.2 Method: CLEAR with POST Data

DELETE /api/hostgroups?clear=true

The post data should be formatted like a POST payload, but the only required field is the host Group Name (name). Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<hostGroups>
  <hostGroup name="hostGroup-100" />
  <hostGroup name="hostGroup-101" />
</hostsGroups>
```

3.4.3 HTTP Status Codes

200 - Hosts were deleted successfully
500 - An internal server error occurred

3.4.4 Example Response

```
<results count='2' success='2' failure='0' entityType='HostGroup' operation='Clear'>
  <result status='success' entity='hostGroup-100'
    location='http://localhost/monitor/api/hostgroups/hostGroup-100' />
<result status='success' entity='hostGroup-101'
    location='http://localhost/monitor/api/hostgroups/hostGroup-101' />
</results>
```

3.5 Host Group Name Autocomplete

Get limited number of sorted existing Host Group names that match a specified prefix. Matching and sorting is done in a case-insensitive fashion. The special wildcard prefix '*' matches all names. Autocomplete names are updated asynchronously when Host Group names are modified.

3.5.1 Method: GET Host Group Name Autocomplete

GET /api/hostgroups/autocomplete/{prefix}

3.5.2 Path Parameters

Field	Type	Description	Required
prefix	Path	Encoded Host Group name prefix to match	yes

3.5.3 HTTP Headers

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

3.5.4 HTTP Status Codes

Code	Description
200	Autocomplete names for prefix returned
404	No autocomplete names for prefix
401	Authentication/authorization error occurred
500	An internal server error occurred while processing autocomplete prefix

3.5.5 Example Responses

Here is an XML example of the autocomplete names returned.

XML results are always wrapped in an `<names>` collection element, with one or more `<name>` subelements.

```
<names>
    <name>HG1</name>
    <name>HG2</name>
</names>
```

Here is a JSON example of the autocomplete names returned.

JSON results are always wrapped in an object with a `names` array member, with one or more name strings.

```
{
  "names" : [ "HG1", "HG2" ]
}
```