2.0 Host APIs

WAS THIS PAGE HELPFUL? Leave Feedback

2.0 Host APIs

2.1 Query Hosts

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

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

2.1.1 Method: GET Hosts

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

2.1.2 Method: GET a single host by name

GET /api/hosts/{hostName}

2.1.3 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
hostName	Path	the name of the host	no
agentId	Query	The agent id managing this host	no
depth	Query	The depth of the response data. Either ‘simple’, ‘shallow’, ‘deep’, ‘sync’, or ‘full’. Defaults to shallow	no

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

2.1.4 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

2.1.5 Query Fields

Field	Description	Alias
id	Host id	hostId
hostName	The name of the host	name host
description	The description of this host	
appType	Application Type name short cut	applicationType.name

applicationType	The associated application type. Valid query fields: applicationType.name applicationType.description applicationType.applicationTypeId applicationType.stateTransitionCriteria	
monitorStatus	The monitor status (from host status)	monitor
agentId	The agent id managing this host	
device	The unique identification for the device associated with this host	deviceIdentification
deviceName	The display name of the device	deviceDisplayName
lastCheckTime	The timestamp for the last host check	
hostGroup	The name of a host group where this host is a member of	
stateType	The state type name	
checkType	The check type name	
properties	one or more valid dynamic property names	

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

Non Queryable Fields

The fields “serviceAvailability” and “bubbleUpStatus” are not queryable since they are calculated fields. However these two fields are calculated and returned in all queries.

For all GET requests, the Hosts Service support three levels of depth:

shallow	primitive data attributes, 1:1 association names, properties (default)
deep	shallow, plus: associated collections (shallow), 1:1 associations shallow
simple	Same as shallow except there are no properties

2.1.6 Example Queries

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

1. *query for all hosts, with shallow depth*
   GET /api/hosts
2. *query for all hosts, with simple depth (only basic attributes, no properties)*
   GET /api/hosts?depth=simple
3. *query for all hosts, with deep depth*
   GET /api/hosts?depth=deep
4. *query for a single host named localhost, with depth of shallow*
   GET /api/hosts/localhost
5. *query for a single host named localhost, with depth of deep*
   GET /api/hosts/localhost?depth=deep
6. *query for a single host named localhost, with depth of simple*
   GET /api/hosts/localhost?depth=simple
7. *query on a single property using LIKE operator*
   GET /api/hosts?query=property.LastPluginOutput like ‘OK%’
8. *query on a property using a range query, and monitor status field, order by a property*
   GET /api/hosts?query=(property.ExecutionTime between 10 and 3500 and (monitorStatus <> 'UP')) order by property.ExecutionTime
9. *query on host’s lastCheckTime using date functions*
   GET /api/hosts?query=day(lastCheckTime) = 22 and month(lastCheckTime) = 5 and minute(lastCheckTime) > 43 order by lastCheckTime
10. *query on two numeric properties using range queries, ordering result*
    GET /api/hosts?query=(property.ExecutionTime < 10 and property.Latency between 800 and 900) order by property.Latency

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

Example Query Results in XML

Here is an XML example of the result of a query finding two hosts at a depth of shallow. Properties are always wrapped in a `<properties>` collection element. All other fields are attributes.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<hosts>
  <host id="1" hostName="localhost" description="localhost" monitorStatus="UP"
    appType="NAGIOS"  deviceIdentification="127.0.0.1"
    lastCheckTime="2013-05-22T09:44:56-07:00"
    bubbleUpStatus="UP" serviceAvailability="9.52" acknowledged="false">
  <properties>
    <property name="MaxAttempts" value="10"/>
    <property name="ExecutionTime" value="6"/>
  </properties>
  <host id="19" hostName="malbec" description="malbec" monitorStatus="UP"
    appType="NAGIOS" deviceIdentification="172.28.113.169"
    lastCheckTime="2013-05-22T09:45:06-07:00"
    bubbleUpStatus="UP" serviceAvailability="0" acknowledged="false">
  </host>
</hosts>
```

For a single host lookup, the <hosts> wrapper is not presented.

```
<host id="19" hostName="malbec" description="malbec" monitorStatus="UP"
  appType="NAGIOS" deviceIdentification="172.28.113.169"
  lastCheckTime="2013-05-22T09:45:06-07:00"
  bubbleUpStatus="UP" serviceAvailability="0" acknowledged="false">
</host>
```

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 Depth Query Response Data - XML - JSON
Deep Depth Query Response Data - XML - JSON
Simple Depth Query Response Data - XML - JSON

2.2 Create Hosts

Post Data AttributCreates a batch (1..n) of new hosts in foundation database. Hosts will be created if they do not exist. If a host exists, it will be updated. Devices will also be created if they do not already exists.

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

2.2.1 Method: POST

POST /api/hosts

2.2.2 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

2.2.3 Request Parameters

Field	Description	Required
merge	Flag that can be used to control host merging for matching host names that differ by case only. Default is true, enabling merging. Valid values are true or false.	False
async	A boolean flag to indicate whether to submit the batch of host updates asynchronously or not. Default is false. Valid values are true or false. A synchronous request will not return back to your client code until the completion of processing all objects provided in the post data. Whereas an asynchronous request will submit the work to a queue, and return immediately.	False

2.2.4 Post Data Attributes

Field	Description	Required
hostName	The name of the host to create. Must be unique.	True
description	A description of the host. Dom Note : If you want a description to show up in Status Viewer, you have to use the Alias property.	False
monitorStatus	The monitor status. Optional, will default to PENDING if not supplied.	False
deviceIdentification	The name of the device for this host. Will be created if it does not already exist.	True
appType	The application type. Must be a valid application type.	True
deviceDisplayName	The name to display in UIs.	Optional
properties	A collection of one or more properties to create with the host	False

2.2.5 XML POST Data Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<hosts>
  <host hostName="host-100" description="First of my servers"
    monitorStatus="UP" appType="NAGIOS"
    deviceIdentification="192.168.5.50" monitorServer="localhost"
    deviceDisplayName="Device-50" >
    <properties>
      <property name="Latency" value="125" />
    </properties>
  </host>
</hosts>
```

More Post Data Examples: XML - JSON

2.2.6 HTTP Status Codes

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

2.2.7 Example Response

In this example, 4 objects were attempted to be created. Only two succeeded, and two failed. The results collection also returns the location of the created or updated host which can be directly used by the GET operation.

```
<results count='4' success='2' failure='2' entityType='Host' operation='Update'>
  <result status='failure' message='Duplicate hostname already exists' entity='host33' />
  <result status='success' entity='host34'
       location='http://localhost/monitor/api/hosts/host34' />
<result status='failure' message='Required field Application Type was not provided'
       entity='host35' />
  <result status='success' entity='host36'
       location='http://localhost/monitor/api/hosts/host36' />
</results>
```

2.3 Delete Host

Deletes one or more hosts from Collage database. There are two methods of deletion:

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

2.3.1 Method: DELETE without POST Data

DELETE /api/hosts/hostName

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

DELETE /api/hosts/hostName1,hostName2
where *hostName1*,*hostName2* is a comma-separated list of hostNames. Note that no-spaces are allowed in this HTTP path segment.

2.3.2 Method: DELETE with POST Data

DELETE /api/hosts

The post data should be formatted like a POST payload, but the only required field is the hostName. Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<hosts>
  <host hostName="host-100" />
  <host hostName="host-101" />
</hosts>
```

JSON, 7.0: (deprecated in 7.1)

```
[
  { "hostName": "host-100" },
  { "hostName": "host-101" }
]
```

JSON 7.1:

```
{ "hosts" : [
  { "hostName": "host-100" },
  { "hostName": "host-101" }
  ]
}
```

2.3.3 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

2.3.4 HTTP Status Codes

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

2.3.5 Example Response

```
<results count='2' success='2' failure='0' entityType='Host' operation='Delete'>
  <result status='success' entity='host-101'
       location='http://localhost/monitor/api/hosts/host-101' />
<result status='success' entity='localhhost-102'
       location='http://localhost/monitor/api/hosts/102' />
</results>
```

2.4 Rename Host

Rename existing Host. Also optionally updates description and Device. Returns shallow version of modified Host.

2.4.1 Method: PUT Rename Host

PUT /api/hosts/rename?oldHostName=old&newHostName=new

2.4.2 HTTP Query Parameters

Field	Type	Description	Required
oldHostName	Query	An encoded existing Host name	yes
newHostName	Query	An encoded new Host name	yes
description	Query	An encoded Host description	no
deviceIdentification	Query	An encoded Device identification to lookup or create	no

2.4.3 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

2.4.4 PUT Post Data

This method accepts no POST data. See HTTP Query Parameters.

2.5.4 HTTP Status Codes

Code	Description
200	Host renamed and returned
404	Host to rename not found
401	Authentication/authorization error occurred
500	An internal server error occurred while processing autocomplete prefix

2.4.5 Example Responses

Shallow Host XML and JSON responses are returned as a single Host element or object as returned by Host queries.

2.5 Host Name Autocomplete

Get limited number of sorted existing Host 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 Hosts host names are modified.

2.5.1 Method: GET Host Name Autocomplete

GET /api/hosts/autocomplete/{prefix}

2.5.2 Path Parameters

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

2.5.3 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

2.5.4 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

2.5.5 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>localhost</name>
    <name>localhost2</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" : [ "localhost", "localhost2" ]
}
```

2.6 Filter Host Groups

Return a list of all hosts at the specified depth filtered by a list of Host Group Names. This filter should be used in place of queries when selecting a large number of hosts by host group criteria. This filter API is designed to run faster than with queries.

2.6.1 Method: GET Hosts BY Host Groups

GET /api/hosts/filter?hostGroupNames={hostGroupNamesList}

2.6.2 HTTP Query and Path Parameters

Field	Type	Description	Required
hostGroupNames	Query	Encoded, comma-separated list of host group names	yes
depth	Query	The depth of the response data. Either ‘shallow’, ‘simple’, or ‘deep’.	Defaults to shallow false

2.6.3 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

2.6.4 Example Filters

1. Retrieve the three hosts groups
   GET /api/hosts/filter?hostGroupNames=hostGroup1,hostGroup2,hostGroup3
2. Retrieve the three hosts groups at depth simple
   GET /api/hosts/filter?hostGroupNames=hostGroup1,hostGroup2,hostGroup3&depth=simple

2.6.5 Example Query Results in XML

Here is an XML example of the result of a filter finding two hosts at a depth of shallow. Properties are always wrapped in a `<properties>` collection element. All other fields are attributes.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<hosts>
  <host id="1" hostName="localhost" description="localhost" monitorStatus="UP"
    appType="NAGIOS"  deviceIdentification="127.0.0.1"
    lastCheckTime="2013-05-22T09:44:56-07:00"
    bubbleUpStatus="UP" serviceAvailability="9.52" acknowledged="false">
  <properties>
    <property name="MaxAttempts" value="10"/>
    <property name="ExecutionTime" value="6"/>
  </properties>
  <host id="19" hostName="malbec" description="malbec" monitorStatus="UP"
    appType="NAGIOS" deviceIdentification="172.28.113.169"
    lastCheckTime="2013-05-22T09:45:06-07:00"
    bubbleUpStatus="UP" serviceAvailability="0" acknowledged="false">
  </host>
</hosts>
```

2.6.6 Appendix data examples

You can use the same data as from section 2.1, the result sets are no different.

See Appendix B and Appendix C for example query data in both XML and JSON:
Shallow Depth Query Response Data - XML - JSON
Deep Depth Query Response Data - XML - JSON
Simple Depth Query Response Data - XML - JSON