Overview

The documentation pages in this section describe the RESTful APIs included with Cascade Gateway products. It is assumed that the reader has practical knowledge of RESTful APIs, so the documentation does not go into detail about what REST is and how to use it. Instead the documentation focuses on what data can be accessed and how to access it.

The following information can be accessed via the API:

- Perform System operations
- Information about system users

Details about REST resources can be found in the Resources section. This overview continues with how to run reports and retrieve data from them.

Authentication

All REST requests must be authenticated. The Authentication section of the Common 1.0 API describes which authentication methods are presently supported. There are also examples that show how to use each of the different authentication methods.

Resources

Ping: Ping

Simple test of service availability.

GET https://{device}/api/gateway/1.1/ping

Authorization

This request requires authorization.

Response Body

On success, the server does not provide any body in the responses.

Users: List users

Get a list of user accounts.

GET https://{device}/api/gateway/1.1/users

Authorization

This request requires authorization.

Response Body

On success, the server returns a response body with the following structure:

JSON
[{
  "authentication_type": string,
  "authorization_type": string,
  "enabled": boolean,
  "first_name": string,
  "id": number,
  "last_access": number,
  "last_authentication": number,
  "last_login": number,
  "last_name": string,
  "login_timeout": number,
  "role": string,
  "username": string,
  "view_packet_details": boolean,
  "view_user_information": boolean
}]

Example:
[
  {
    "username": "admin",
    "last_authentication": 1352313328,
    "first_name": "Jonh",
    "last_name": "Smith",
    "authorization_type": "Local",
    "enabled": true,
    "view_user_information": true,
    "authentication_type": "Local",
    "role": "Administrator",
    "login_timeout": 900,
    "last_login": 1352313328,
    "last_access": 1352313328,
    "id": 123
  },
  {
    "username": "admin2",
    "last_authentication": 1352313328,
    "first_name": "Mark",
    "last_name": "Greg",
    "authorization_type": "Local",
    "enabled": true,
    "view_user_information": true,
    "authentication_type": "Local",
    "role": "Administrator",
    "login_timeout": 900,
    "last_login": 1352313328,
    "last_access": 1352313328,
    "id": 124
  }
]

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>users</td>
<td>&lt;array of &lt;object&gt;&gt;</td>
<td>List of user accounts on the system.</td>
<td></td>
</tr>
<tr>
<td>users[user]</td>
<td>&lt;object&gt;</td>
<td>User account.</td>
<td></td>
</tr>
<tr>
<td>users[user].authentication_type</td>
<td>&lt;string&gt;</td>
<td>Type of authentication for the user, such as Local or RADIUS.</td>
<td>Values: Local, RADIUS</td>
</tr>
<tr>
<td>users[user].authorization_type</td>
<td>&lt;string&gt;</td>
<td>Type of authorization for the user, such as Local or RADIUS.</td>
<td>Values: Local, RADIUS</td>
</tr>
<tr>
<td>users[user].enabled</td>
<td>&lt;boolean&gt;</td>
<td>Boolean flag indicating if the user account is enabled.</td>
<td></td>
</tr>
<tr>
<td>users[user].first_name</td>
<td>&lt;string&gt;</td>
<td>First name of the user.</td>
<td></td>
</tr>
<tr>
<td>users[user].id</td>
<td>&lt;number&gt;</td>
<td>Numeric ID of the user that the system uses internally and in the API.</td>
<td></td>
</tr>
<tr>
<td>users[user].last_access</td>
<td>&lt;number&gt;</td>
<td>Time of last access to the system. Unix time (epoch).</td>
<td></td>
</tr>
<tr>
<td>users[user].last_authentication</td>
<td>&lt;number&gt;</td>
<td>Time of last authentication. Unix time (epoch).</td>
<td></td>
</tr>
<tr>
<td>users[user].last_login</td>
<td>&lt;number&gt;</td>
<td>Time of last login. Unix time (epoch).</td>
<td></td>
</tr>
<tr>
<td>users[user].last_name</td>
<td>&lt;string&gt;</td>
<td>Last name of the user.</td>
<td></td>
</tr>
<tr>
<td>users[user].login_timeout</td>
<td>&lt;number&gt;</td>
<td>Timeout (in seconds) during which the user cannot log in to the system because of security policies.</td>
<td></td>
</tr>
<tr>
<td>users[user].role</td>
<td>&lt;string&gt;</td>
<td>Role of the user. Defines permissions.</td>
<td>Values: Developer, Administrator, Operator, Monitor, Event_Viewer, Dashboard_Viewer</td>
</tr>
<tr>
<td>users[user].username</td>
<td>&lt;string&gt;</td>
<td>User name (short name) that identifies the user to the system, such as 'admin'.</td>
<td></td>
</tr>
</tbody>
</table>
Users: Test RADIUS user

Test a RADIUS user.

```
GET https://{device}/api/gateway/1.1/users/radius/test_user?username={string}&password={string}
```

Authorization
This request requires authorization.

Parameters

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>&lt;string&gt;</td>
<td>RADIUS username.</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>&lt;string&gt;</td>
<td>RADIUS username password.</td>
<td></td>
</tr>
</tbody>
</table>

Response Body

On success, the server returns a response body with the following structure:

```
JSON
```
Example:
{
  "authenticated": true,
  "permission": ",",
  "permission_id": ",",
  "error_message": ",",
  "role_id": 0,
  "role": ",",
  "authorized": false,
  "server_ip": "10.38.8.112:1812",
  "attributes": [
    {"25": "operatorClass"},
    {"25": "monitorClass"},
    {"25": "eventviewerClass"},
    {"17164": "unMappedRole"},
    {"17164": "monitorCascade"},
    {"17164": "eventviewerCascade"},
    {"17164": "dashboardCascade"},
    {"25": "DBAccess"},
    {"25": "dashboardClass"},
    {"17164": "AbC10~!@#$%^&*()_+{}|\[":;<>?/.'z"},
    {"17164": "operatorCascade"},
    {"LOGIN_RADIUS_SERVER": "10.38.8.112:1812"},
    {"25": "adminClass1"},
    {"25": "unMappedClass"},
    {"25": "eventviewerClass"},
    {"17164": "adminCascade"}],
  "details": "Using 10.38.8.112:1812 - Unable to match a role."
}
### Users: Test RADIUS server
Test the connection to a RADIUS server.

**GET** `https://{device}/api/gateway/1.1/users/radius/test_server`

**Authorization**
This request requires authorization.

**Parameters**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>&lt;string&gt;</td>
<td>RADIUS server identifier, example server=IP:PORT.</td>
<td></td>
</tr>
</tbody>
</table>

**Response Body**
On success, the server returns a response body with the following structure:

```
{
  "message": string,
  "success": boolean
}
```

Example:
```
{
  "message": "RADIUS connection attempt succeeded",
  "success": true
}
```

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_server</td>
<td>&lt;object&gt;</td>
<td>RadiusTestServerResponse object.</td>
<td></td>
</tr>
<tr>
<td>test_server.message</td>
<td>&lt;string&gt;</td>
<td>Response message.</td>
<td></td>
</tr>
<tr>
<td>test_server.success</td>
<td>&lt;boolean&gt;</td>
<td>Flag indication if the RADIUS server test was successful.</td>
<td></td>
</tr>
</tbody>
</table>

### Users: Re-authenticate user
Re-authenticate a user account. Requires basic authentication.

**GET** `https://{device}/api/gateway/1.1/users/re_authenticate`

**Authorization**
This request requires authorization.

**Response Body**
On success, the server does not provide any body in the responses.
Users: Get user

Get a user account by user ID.

GET https://{device}/api/gateway/1.1/users/{user_id}

Authorization

This request requires authorization.

Response Body

On success, the server returns a response body with the following structure:

```json
{
  "username": "admin",
  "last_authentication": 1352313328,
  "first_name": "John",
  "last_name": "Smith",
  "authorization_type": "Local",
  "enabled": true,
  "view_user_information": true,
  "authentication_type": "Local",
  "role": "Administrator",
  "login_timeout": 900,
  "last_login": 1352313328,
  "last_access": 1352313328,
  "id": 123
}
```

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>&lt;object&gt;</td>
<td>User account.</td>
<td></td>
</tr>
<tr>
<td>user.authentication_type</td>
<td>&lt;string&gt;</td>
<td>Type of authentication for the user, such as Local or RADIUS.</td>
<td>Values: Local, RADIUS</td>
</tr>
<tr>
<td>user.authorization_type</td>
<td>&lt;string&gt;</td>
<td>Type of authorization for the user, such as Local or RADIUS.</td>
<td>Values: Local, RADIUS</td>
</tr>
<tr>
<td>user.enabled</td>
<td>&lt;boolean&gt;</td>
<td>Boolean flag indicating if the user account is enabled.</td>
<td></td>
</tr>
<tr>
<td>user.first_name</td>
<td>&lt;string&gt;</td>
<td>First name of the user.</td>
<td></td>
</tr>
<tr>
<td>user.id</td>
<td>&lt;number&gt;</td>
<td>Numeric ID of the user that the system uses internally and in the API.</td>
<td></td>
</tr>
<tr>
<td>user.last_access</td>
<td>&lt;number&gt;</td>
<td>Time of last access to the system. Unix time (epoch).</td>
<td></td>
</tr>
<tr>
<td>user.last_authentication</td>
<td>&lt;number&gt;</td>
<td>Time of last authentication. Unix time (epoch).</td>
<td></td>
</tr>
<tr>
<td>user.last_login</td>
<td>&lt;number&gt;</td>
<td>Time of last login. Unix time (epoch).</td>
<td></td>
</tr>
<tr>
<td>user.last_name</td>
<td>&lt;string&gt;</td>
<td>Last name of the user.</td>
<td></td>
</tr>
<tr>
<td>user.login_timeout</td>
<td>&lt;number&gt;</td>
<td>Timeout (in seconds) during which the user cannot log in to the system</td>
<td>Values: Developer, Administrator, Operator, Monitor, Event_Viewer, Dashboard_Viewer</td>
</tr>
<tr>
<td>user.role</td>
<td>&lt;string&gt;</td>
<td>Role of the user. Defines permissions.</td>
<td></td>
</tr>
<tr>
<td>user.username</td>
<td>&lt;string&gt;</td>
<td>User name (short name) that identifies the user to the system, such as 'admin'.</td>
<td></td>
</tr>
<tr>
<td>user.view_packet_details</td>
<td>&lt;boolean&gt;</td>
<td>Boolean flag indicating if the user has access to packet data.</td>
<td>Optional</td>
</tr>
<tr>
<td>user.view_user_information</td>
<td>&lt;boolean&gt;</td>
<td>Boolean flag indicating if the user has access to identity information, such as Active Directory information.</td>
<td>Optional</td>
</tr>
</tbody>
</table>
System: Stop all processes
Stop all system processes. The operation is asynchronous. Use "GET system/status" to poll for status. On Enterprise systems, stop system processes on all modules.

POST https://{device}/api/gateway/1.1/system/stop

Authorization
This request requires authorization.

Request Body
Do not provide a request body.

Response Body
On success, the server does not provide any body in the responses.

System: Kill all processes
Kill all system processes. The operation is asynchronous. Use "GET system/status" to poll for status. On Enterprise systems, kill system processes on all modules. Warning: this operation can result in data being corrupted.

POST https://{device}/api/gateway/1.1/system/kill

Authorization
This request requires authorization.

Request Body
Do not provide a request body.

Response Body
On success, the server does not provide any body in the responses.

System: Shutdown
Shutdown the system. The operation is asynchronous.

POST https://{device}/api/gateway/1.1/system/shutdown

Authorization
This request requires authorization.

Request Body
Do not provide a request body.

Response Body
On success, the server does not provide any body in the responses.

System: Get status of all processes
Get status of all system processes. On Enterprise systems, get system process statuses on all modules.

GET https://{device}/api/gateway/1.1/system/status

Authorization
This request requires authorization.

Response Body
On success, the server returns a response body with the following structure:

JSON
[ {
  "module_ipaddr": string,
  "module_name": string,
  "process_id": string,
  "process_name": string,
  "status": string
}
]

Example:
[
  {
    "process_id": "25096",
    "process_name": "memmonitor",
    "status": "Running"
  },
  {
    "process_name": "healthd",
    "status": "Stopped"
  },
  {
    "process_id": "25092",
    "process_name": "diskmon",
    "status": "Running"
  },
  {
    "process_id": "25123",
    "process_name": "dispatcher",
    "status": "Running"
  },
  {
    "process_name": "analyzer",
    "status": "Stopped"
  }
]

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>&lt;array of &lt;object&gt;&gt;</td>
<td>SystemStatus object.</td>
<td></td>
</tr>
<tr>
<td>status[process]</td>
<td>&lt;object&gt;</td>
<td>SystemProcess object.</td>
<td></td>
</tr>
<tr>
<td>status[process].module_ipaddr</td>
<td>&lt;string&gt;</td>
<td>Module IP Address. Available on Enterprise systems only.</td>
<td>Optional</td>
</tr>
<tr>
<td>status[process].module_name</td>
<td>&lt;string&gt;</td>
<td>Module name. Available on Enterprise systems only.</td>
<td>Optional</td>
</tr>
<tr>
<td>status[process].process_id</td>
<td>&lt;string&gt;</td>
<td>Process ID.</td>
<td>Optional</td>
</tr>
<tr>
<td>status[process].process_name</td>
<td>&lt;string&gt;</td>
<td>Process name.</td>
<td></td>
</tr>
<tr>
<td>status[process].status</td>
<td>&lt;string&gt;</td>
<td>Process status.</td>
<td>Values: Running, Stopped</td>
</tr>
</tbody>
</table>

**System: Start all processes**

Start all system processes. The operation is asynchronous. Use "GET system/status" to poll for status. On Enterprise systems, start system processes on all modules.

POST https://{device}/api/gateway/1.1/system/start

**Authorization**

This request requires authorization.

**Request Body**

Do not provide a request body.

**Response Body**

On success, the server does not provide any body in the responses.

**System: Restart all processes**

Restart all system processes. The operation is asynchronous. Use "GET system/status" to poll for status. On Enterprise systems, stop system processes on all modules.

POST https://{device}/api/gateway/1.1/system/restart
Authorization
This request requires authorization.

Request Body
Do not provide a request body.

Response Body
On success, the server does not provide any body in the responses.

---

System: Reboot
Reboot the system. The operation is asynchronous.

```
POST https://{device}/api/gateway/1.1/system/reboot
```

Authorization
This request requires authorization.

Request Body
Do not provide a request body.

Response Body
On success, the server does not provide any body in the responses.

---

System: Stop all processes (one module)
Stop all system processes on one module on Enterprise systems. The operation is asynchronous. Use "GET system/{module}/status" to poll for status. The {module} can be either the IP Address or the module name.

```
POST https://{device}/api/gateway/1.1/system/{module}/stop
```

Authorization
This request requires authorization.

Request Body
Do not provide a request body.

Response Body
On success, the server does not provide any body in the responses.

---

System: Kill all processes (one module)
Kill all system processes on one module on Enterprise systems. The operation is asynchronous. Use "GET system/{module}/status" to poll for status. The {module} can be either the IP Address or the module name. Warning: This operation can result in data being corrupted.

```
POST https://{device}/api/gateway/1.1/system/{module}/kill
```

Authorization
This request requires authorization.

Request Body
Do not provide a request body.

Response Body
On success, the server does not provide any body in the responses.

---

System: Get status of all processes (one module)
Get status of all system processes on one module on Enterprise systems. The {module} can be either the IP
Address or the module name.

GET https://{device}/api/gateway/1.1/system/{module}/status

Authorization
This request requires authorization.

Response Body
On success, the server returns a response body with the following structure:

```json
[
  {
    "module_ipaddr": "string",
    "module_name": "string",
    "process_id": "string",
    "process_name": "string",
    "status": "string"
  }
]
```

Example:
```
[
  {
    "process_id": "25096",
    "process_name": "memmonitor",
    "status": "Running"
  },
  {
    "process_name": "healthd",
    "status": "Stopped"
  },
  {
    "process_id": "25092",
    "process_name": "diskmon",
    "status": "Running"
  },
  {
    "process_id": "25123",
    "process_name": "dispatcher",
    "status": "Running"
  },
  {
    "process_name": "analyzer",
    "status": "Stopped"
  }
]
```

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>&lt;array of &lt;object&gt;&gt;</td>
<td>SystemStatus object.</td>
<td></td>
</tr>
<tr>
<td>status[process]</td>
<td>&lt;object&gt;</td>
<td>SystemProcess object.</td>
<td></td>
</tr>
<tr>
<td>status[process].module_ipaddr</td>
<td>&lt;string&gt;</td>
<td>Module IP Address. Available on Enterprise systems only.</td>
<td>Optional</td>
</tr>
<tr>
<td>status[process].module_name</td>
<td>&lt;string&gt;</td>
<td>Module name. Available on Enterprise systems only.</td>
<td>Optional</td>
</tr>
<tr>
<td>status[process].process_id</td>
<td>&lt;string&gt;</td>
<td>Process ID.</td>
<td>Optional</td>
</tr>
<tr>
<td>status[process].process_name</td>
<td>&lt;string&gt;</td>
<td>Process name.</td>
<td></td>
</tr>
<tr>
<td>status[process].status</td>
<td>&lt;string&gt;</td>
<td>Process status.</td>
<td>Values: Running, Stopped</td>
</tr>
</tbody>
</table>

System: Start all processes (one module)
Start all system processes on one module on Enterprise systems. The operation is asynchronous. Use “GET system/{module}/status” to poll for status. The {module} can be either the IP Address or the module name.

POST https://{device}/api/gateway/1.1/system/{module}/start

Authorization
This request requires authorization.

Request Body
Do not provide a request body.
Response Body
On success, the server does not provide any body in the responses.

System: Restart all processes (one module)
Restart all system processes on one module on Enterprise systems. The operation is asynchronous. Use "GET system/{module}/status" to poll for status. The {module} can be either the IP Address or the module name.

POST https://{device}/api/gateway/1.1/system/{module}/restart

Authorization
This request requires authorization.

Request Body
Do not provide a request body.

Response Body
On success, the server does not provide any body in the responses.

Error Codes
In the event that an error occurs while processing a request, the server will respond with appropriate HTTP status code and additional information in the response body:

```
{
  "error_id": "{error identifier}",
  "error_text": "{error description}",
  "error_info": {error specific data structure, optional}
}
```

The table below lists the possible errors and the associated HTTP status codes that may returned:

<table>
<thead>
<tr>
<th>Error ID</th>
<th>HTTP Status</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERNAL_ERROR</td>
<td>500</td>
<td>Internal server error.</td>
</tr>
<tr>
<td>AUTH_REQUIRED</td>
<td>401</td>
<td>The requested resource requires authentication.</td>
</tr>
<tr>
<td>AUTH_INVALID_CREDENTIALS</td>
<td>401</td>
<td>Invalid username and/or password.</td>
</tr>
<tr>
<td>AUTH_INVALID_SESSION</td>
<td>401</td>
<td>Session ID is invalid.</td>
</tr>
<tr>
<td>AUTH_EXPIRED_PASSWORD</td>
<td>403</td>
<td>The password must be changed. Access only to password change resources.</td>
</tr>
<tr>
<td>AUTH_DISABLED_ACCOUNT</td>
<td>403</td>
<td>Account is either temporarily or permanently disabled.</td>
</tr>
<tr>
<td>AUTH_FORBIDDEN</td>
<td>403</td>
<td>User is not authorized to access the requested resource.</td>
</tr>
<tr>
<td>AUTH_INVALID_TOKEN</td>
<td>401</td>
<td>OAuth access token is invalid.</td>
</tr>
<tr>
<td>AUTH_EXPIRED_TOKEN</td>
<td>401</td>
<td>OAuth access token is expired.</td>
</tr>
<tr>
<td>AUTH_INVALID_CODE</td>
<td>401</td>
<td>OAuth access code is invalid.</td>
</tr>
<tr>
<td>AUTH_EXPIRED_CODE</td>
<td>401</td>
<td>OAuth access code is expired.</td>
</tr>
<tr>
<td>RESOURCE_NOT_FOUND</td>
<td>404</td>
<td>Requested resource was not found.</td>
</tr>
<tr>
<td>HTTP_INVALID_METHOD</td>
<td>405</td>
<td>Requested method is not available for this resource.</td>
</tr>
<tr>
<td>HTTP_INVALID_HEADER</td>
<td>400</td>
<td>An HTTP header was malformed.</td>
</tr>
<tr>
<td>REQUEST_INVALID_INPUT</td>
<td>400</td>
<td>Malformed input structure.</td>
</tr>
<tr>
<td>URI_INVALID_PARAMETER</td>
<td>400</td>
<td>URI parameter is not supported or malformed.</td>
</tr>
<tr>
<td>URI_MISSING_PARAMETER</td>
<td>400</td>
<td>Missing required parameter.</td>
</tr>
</tbody>
</table>