



Using the ScienceLogic API

Skylar One version 12.5.20

Table of Contents

Introduction to the ScienceLogic API	9
What is the ScienceLogic API?	10
Accessing the API	10
API Settings	12
HTTP Methods, Headers and Response Formats	14
HTTP Methods	15
HTTP Status Codes	16
Skylar One-Specific Headers	17
Response Headers	17
Request Headers	17
Response Formats	18
Resources & URIs	19
Available Resources	20
URI Formatting	21
Resource Index Responses	22
Constructing URIs Using a searchspec	23
Filters	24
Options	25
Sorting	26
Specifying a Query String in the Request Body	26
Required Options for Indexes	27
Resource Responses	27
Creating and Updating Resources	28
Asynchronous Operations	29
Links Between Resources	29
Size Limits	30
Authentication and Access Permissions	31
User Access to the API	32
API Keys	32
Account Lockouts	32
The _self Resource	33

Audit Logging	34
Using API Keys in Skylar One	35
How API Keys Work in Skylar One	36
Best Practices for API Keys	36
Enabling API Keys in Skylar One	37
Enforcing a Minimum Expiration Window for API Keys	37
Assigning a User Policy for API Key Accounts	38
Managing API Keys in GraphQL	38
Creating an API Key in GraphQL	38
Deleting an API Key in GraphQL	40
Querying API Keys	41
Managing API Keys in Skylar One	42
Viewing the List of API Keys	42
Creating an API Key in the Skylar One User Interface	43
Viewing API Key Details	44
Regenerating an API Key	44
Deleting an API Key	45
Cleaning Up Expired Keys	45
Using an API Key to Authenticate API Requests	45
Custom Attributes	47
Custom Attributes for API Resources	48
Viewing and Adding Custom Attributes	48
Example of How to Add Custom Attributes	51
Editing Custom Attributes	53
Requests to Resources with Custom Attributes	54
Removing Custom Attributes	56
Generating Events Using the API	57
Generating Alerts	58
Defining API Event Policies	59
Defining API Event Policies in the Classic Skylar One User Interface	61
Requesting Performance Data in Bulk	63
Resource URIs	64

Specifying the Time Range for a Data Request	66
Specifying Data Fields	66
Fields for Dynamic Application Resources	67
Fields for Port Monitor Resources	67
Fields for Web Content Monitor Resources	68
Fields for SOAP/XML Transaction Monitor Resources	70
Fields for Process Monitor Resources	71
Fields for Windows Service Monitor Resources	72
Fields for Email Round-Trip Monitor Resources	73
Fields for DNS Monitor Resources	74
Fields for File System Resources	75
Fields for Availability Resources	76
Fields for Interface Resources	77
Fields for CBQoS Resources	85
Requesting Data for Specific Devices or Interfaces	90
Filtering Device Resources	91
Filtering Interface Resources	92
Filtering CBQoS Resources	94
Additional Options	94
Responses from Bulk Performance Data Resources	95
Best Practices for Requesting Bulk Performance Data	96
Using the Ticket Resource	98
Requirements	100
Getting Started	100
Connecting to the API	101
Viewing a List of Tickets	107
Viewing a List of Tickets and Ticket Details	114
Filtering a List of Tickets	116
Retrieving Information about a Specific Ticket	117
Updating a Ticket	120
Capture Ticket Information in a File	120
Edit the Captured File	122

Use HTTP POST to Update the Ticket with the Edited File	123
Sending Only Changes in the ticket99.json File	126
Creating a New Ticket	126
Capturing an Existing Ticket and Storing the Information in a File	127
Determining the URI for a User Account	127
Editing the Captured File	132
Using the Edited File to Create a New Ticket	134
Viewing Notes for a Ticket	137
Adding a Note to a Ticket	141
Capturing an Existing Note and Storing the Information in a File	142
Editing the Captured File	142
Creating a New Note Using the Edited File	143
Viewing the Attachments for a Ticket	144
Adding an Attachment to a Ticket Note	150
Using the Discovery Resource	152
Requirements	154
Getting Started	154
Connecting to the API	155
Viewing a List of Discovery Sessions	159
Viewing Details about All Discovery Sessions	166
Filtering the List of Discovery Sessions	168
Retrieving Information about a Specific Discovery Session	169
Starting a Discovery Session	171
Viewing a List of All Active Discovery Sessions	173
Retrieving Information about a Specific Active Discovery-Session	175
Viewing Logs for a Discovery Session	176
Stopping a Currently Running Discovery-Session	179
Deleting a Discovery Session	181
Searching Component Trees	182
Searching for All the Components in a Tree	183
Searching for the Direct Children of a Device	184
Searching for the Components in a Sub-Tree	185

Searching for a Component by Unique ID	189
Simple Provisioning System	193
System Design	195
Prerequisites	196
System-Specific Functions	197
Utility Functions (utils.php)	198
Performing Requests	198
Requesting a List of Entities	204
Organization Lookup	207
Creating Entities	208
Deleting Entities	209
Configuring SNMP Credentials	211
Requesting Discovery Session Logs	216
Requesting an Available Data Collection Unit	222
Requesting a List of Referenced Entities	225
User Interface	228
header.php	228
index.php	229
devices.php	230
remove.php	235
provisioning.css	236
Provisioning a Customer (provision_customer.php)	236
Retrieving and Configuring Devices (configure_devices.php)	244
Removing a Customer (delete_customer.php)	257
Create Device Maintenance Schedules via the API	262
Requirements	263
Caveats to Consider	263
Prerequisite Examples	263
Getting Started	265
Creating the Task (Step 1)	265
Creating the Schedule Entry (Step 2)	267
Aligning the Task to the Schedule Entry (Step 3)	269

Available Actions	271
Accounts	274
Account Lockouts	274
Alerts	275
Appliances	275
Assets	275
CBQoS Metrics	278
CBQoS Objects	278
CBQoS Object Types	278
Cleared Events	278
Collection Labels	279
Collection Label Groups	279
Collector Groups	279
Credentials	280
Custom Attributes	282
Dashboards	284
Devices	285
Device Categories	288
Device Classes	288
Device Groups	288
Device Relationships	289
Device Relationship Types	289
Device Templates	290
Discovery Sessions	292
Dynamic Applications	293
Events	308
Event Categories	308
External Contacts	308
File Uploads	309
Interfaces	309
Interface Metrics	310
Interface Tags	310

Monitors	311
Organizations	313
Performance Data	314
PowerPacks	316
Product SKUs	316
Scale Values	317
Schedules	317
Streamer Push Proxy	317
System Patches	318
System Settings	319
System Thresholds	319
Tasks	319
Themes	320
Threshold Overrides	320
Tickets	321
Ticket Categories	322
Ticket Chargeback	323
Ticket Logs	323
Ticket Notes	323
Ticket Queues	324
Ticket States	324
Unit Values	325
User Policies	325
Vendors	325

Chapter

1

Introduction to the ScienceLogic API

Overview

This manual describes the functionality of the ScienceLogic API and is intended for developers who are responsible for integrating Skylar One with external systems. To use this manual, you should have a general understanding of the HTTP protocol.

This chapter covers the following topics:

<i>What is the ScienceLogic API?</i>	10
<i>Accessing the API</i>	10
<i>API Settings</i>	12

What is the ScienceLogic API?

The ScienceLogic API allows external systems to programmatically access data in Skylar One. The API gives access to entities in Skylar One— such as tickets, devices, and collected data – using standard HTTP request/response protocols. Much like the user interface provides access to Skylar One for end users, the API provides access to Skylar One for external systems.

The following Skylar One appliances provide access to the API:

- All-In-One Appliances
- Administration Portals
- Database Servers

Accessing the API

This section gives a brief overview of how to communicate with an appliance that provides access to the API. All communication with the API is handled by HTTPS requests.

A request must include:

- **Valid Skylar One login credentials.** The API uses HTTP authentication methods. The credentials you include in the HTTP request are validated against the user accounts stored in the system.
- **A Resource URI.** The URI for the resource (entity) you are performing the request on.
- **An HTTP Method.** Correlates to the action you would like to perform on the resource.
- **An Accept Header.** Specifies which format should be used for the response. The API supports application/xml and application/json formats.
- **The base URI of the API.** The base URI of the API is the full address of the main API index. The base URI includes information about the appliance you are using to access the API:

- For Database Servers, Administration Portals, and All-In-One Appliances, the base URI of the API is:

```
https://<ip-address or hostname of appliance>/api
```

- For Skylar One PowerFlow, the base URI of the API is:

```
https://<ip-address or hostname of appliance>
```

The response from the API contains:

- **An HTTP Status Code.** Indicates the result of the request.
- **Skylar One-Specific Status Headers.** Contains additional information about the result of a request. This information supplements the HTTP Status Code.
- **XML or JSON data.** Information about the requested resource in the format specified in the request.

To familiarize yourself with performing basic requests, you can use a standard web browser:

1. Open a web browser.

NOTE: When you request a resource from the API using a web browser, the API will respond in raw XML format. Some browsers, including Safari and Internet Explorer, will not display raw XML correctly. If possible, you should use Mozilla Firefox to perform these steps.

2. Navigate to the base URI of the API for the appliance you are using. The standard authentication window is displayed.
3. Enter the username and password for a user account in the system. The response for the main resource index is displayed.

The response contains a list of URIs for the resources that are available through the API:

Each entry in the list includes:

- The URI of the resource.
- A description of the resource.

NOTE: If you are accessing the API through an Administration Portal, Database Server, or All-In-One Appliance, the "/api" portion of the base URI is included in all resource URIs returned by the API.

For example, the URI "/organization" has the description "Get/Update/Add/Delete Organizations". To view information about organizations, append the base URI of the API with the URI for the organization resource:

```
<base URI>/organization
```

The index for the organization resource, which contains descriptions and URIs for every organization in the system, is returned.

The browser handles the required elements of the request in the following ways:

- The credentials you enter are used to authenticate the request. Most browsers will save these credentials so you need to enter them only once per session.
- You enter the resource URI (/organization) in the browser address bar.
- By default, the browser performs a GET request.

- The browser supplies a default accept header with the request. The default accept header used by Mozilla Firefox contains "application/xml", one of the two response formats returned by the API.

Although using a web browser is the easiest way to make simple requests, using a browser provides limited control and functionality. For example, you cannot explicitly perform PUT, POST or DELETE requests with a browser. A browser will also handle certain aspects of requests and responses, such as automatically following redirects, differently than they will be handled by integration code. ScienceLogic recommends you use command line cURL to test requests.

API Settings

The **REST API Settings** page System > Settings > API allows you to define global parameters that affect the behavior of the ScienceLogic API.

NOTE: The **REST API Settings** page is available only to administrator users.

To define or edit the settings in the **REST API Settings** page:

1. Go to the **REST API Settings** page System > Settings > API.
2. In the **REST API Settings** page, edit the values in one or more of the following fields:
 - **Internal Request Account.** Specify the user account that allows appliances to make API requests without a password.
 - **X-EM7-run-as Header Support.** Specifies whether administrator users can make API requests using the permissions of another user without that user's password. Choices are
 - *Disabled.* Administrator users cannot make API requests using the permissions of another user.
 - *Enabled (Admin only).* Administrator users can include the X-EM7-run-as Header to make API requests using the permissions of another user.
 - **Logging.** Specifies which logs Skylar One will write to when tickets are created or updated using the API. Choices are:
 - *Transaction Logging Only (System Logs).* If a ticket is created or updated using the API, Skylar One will write an entry to the audit log that indicates that a user performed a write-operation using the API. However, Skylar One will not write to the ticket log for the ticket that was created or updated.
 - *Normal (Ticket and System Logs).* If a ticket is created or updated using the API, Skylar One will write to the audit log and to the ticket log for the ticket that was created or updated.
 - **X-EM7-suppress-logging Header Support.** If *Normal (Ticket and System Logs)* is selected in the **Logging** field, this field specifies whether an administrator can use the X-EM7-suppress-logging header can be used when creating or updating a ticket with the API. If the X-EM7-suppress-logging header is used when creating or updating a ticket, Skylar One will not write to the ticket log for that ticket.

- *Disabled.* The X-EM7-suppress-logging header cannot be used.
 - *Enabled (Admin only).* The X-EM7-suppress-logging header can be used to stop Skylar One from writing to the ticket log for the ticket that was created or updated.
 - **Send Notification.** When a ticket is created or updated, Skylar One can automatically send notification Emails to the ticket assignee and ticket watchers. This option specifies the conditions under which Skylar One will send notification Emails when tickets are created or updated using the API. Choices are:
 - *Only if X-EM7-send-notification:1 is sent.* EM7 will send notification Emails for a ticket only when the X-EM7-send-notification header is set to 1.
 - *Sent after every write operation.* Skylar One will send notification Emails for every API request that creates or updates a ticket.
3. Select the **[Save]** button to save changes in this page.

Chapter

2

HTTP Methods, Headers and Response Formats

Overview

This chapter covers how the API uses elements of the HTTP protocol to handle and respond to requests.

This chapter covers the following topics:

<i>HTTP Methods</i>	15
<i>HTTP Status Codes</i>	16
<i>Skylar One-Specific Headers</i>	17
<i>Response Formats</i>	18

HTTP Methods

To perform operations on API resources, you can use one of the following four HTTP methods in your requests. Each resource has a different set of rules that determines which of the four methods can be used to make requests.

GET

GET fetches resources. The response to a GET request contains information about the resource you requested.

IMPORTANT: ScienceLogic recommends that you include an order parameter (e.g. {order._id:'ASC'}) to the request when making multiple calls to ensure consistent data in each GET request from `api/device/`.

POST

POST updates an existing resource or creates a new resource:

- To update a resource, use the POST method in a request to a specific instance of a resource. For example, to update a ticket with ID "1", you would POST JSON or XML data to the following URI:

```
/ticket/1
```

If POST is used to update a resource, not all attributes of the resource need to be specified. The API will update only the attributes specified in the request.

- To create a resource, use the POST method in a request to the index for that resource. For example, to create a ticket, you would POST JSON or XML data to the following URI:

```
/ticket
```

The system creates a unique ID for the new resource. The URI for the new resource is based on the unique ID for the resource.

PUT

PUT adds or replaces a resource. Unlike POST, PUT will replace an entire resource. PUT requires a specific resource URI. The result of a PUT request will be consistent if the request is repeated.

DELETE

DELETE removes resources. If a resource allows the DELETE method, a successful DELETE request will remove the corresponding entry in the ScienceLogic Database.

HTTP Status Codes

The API uses standard HTTP status codes to indicate the general result of a request. Every response from the API will have one of the following status codes in the header:

- **200 OK.** Indicates that the request was valid and the transaction executed normally.
- **201 Created.** Indicates that a new resource was created. **201 Created** is *not* used when a resource is updated.
- **202 Accepted.** Indicates the request was accepted for processing.
- **204 No Content Returned.** Indicates the request was successful but the API returned no content. This response is typical when a file is uploaded via a PUT request.
- **301 Moved Permanently.** Indicates that the request was made on a sub-resource, but the sub-resource ID used in the request URI does not match a sub-resource associated with the main resource. For example, a GET request was made for an interface (the sub-resource) for a device (the main resource), but the interface ID in the URI is associated with a different device than the device ID used in the URI.
- **302 Found.** Indicates that the request did not include required options or filters. If a response has a **302 Found** status code, a "Location" header will be included in the response. The "Location" header will contain the URI of your request with the default required options or filters included.
- **303 See Other.** Indicates that the request is not the preferred means of fetching the resource. If a response has a **303 See Other** status code, a "Location" header will be included in the response. The "Location" header will contain the URI for the preferred means of fetching the resource.
- **400 Bad Request.** Indicates that the XML or JSON posted with the request contained bad syntax or was missing required fields.
- **401 Unauthorized.** Indicates invalid credentials were provided for authentication.
- **403 Forbidden.** Indicates that the credentials provided for authentication were valid, but the user is not permitted to access the resource.
- **404 Not Found.** Indicates that there is no resource at the URI specified in the request.
- **405 Method Not Allowed.** Indicates that the method used in the request is not permitted with the specified resource. For example, the DELETE method cannot be used on a ticket resource.
- **406 Method Not Acceptable.** Indicates that the accept header included in the request does not allow an XML or JSON response.
- **415 Unsupported Media Type.** Indicates that the content-type provided in a PUT or POST request is not supported.
- **500 Internal Server Error.** Indicates that a general error has occurred with the request that is not described by another status code. The **X-EM7-Status-Message** header may contain more information.
- **501 Not Implemented.** Indicates that the requested resource is a placeholder for future use.

Skylar One-Specific Headers

Response Headers

In addition to HTTP status codes, every response from the API includes headers that provide additional details about the result of a request:

- ***X-EM7-Implemented-methods***. A comma-delimited list of methods that are supported by the requested resource. This header is intended to provide information on the actions that can be performed on a given resource. For example, if you perform a GET request on the `/device` resource index, ***X-EM7-Implemented-methods*** will contain "GET,POST", the two methods supported by `/device`. If you perform a GET request on a specific device (e.g. `/device/1`), the ***X-EM7-Implemented-methods*** header will contain "GET,POST,PUT,DELETE", because a specific device resource supports all available methods.
- ***X-EM7-Applicable-resources***. A comma-delimited list of base URIs for resources that can be applied to the requested resource. For example, to start a discovery session through the API, you would POST a specific `/discovery_session` resource to the `/discovery_session_active` resource index; therefore, if you perform a GET request on the `/discovery_session_active` resource index, the response will include a ***X-EM7-Applicable-resources*** header of `/discovery_session`. For more information on applying resource URIs to other resources, see the [Asynchronous Operations](#) section.
- ***X-EM7-authenticated-user***. The URI of the user account that authenticated the request. If the request included the `X-EM7-run-as` header, the `X-EM7-authenticated-user` will return the run-as user.
- ***X-EM7-status-code***. Typically a human-readable version of the HTTP Status Code. For certain errors, ***X-EM7-status-code*** might include additional information about why a request was unsuccessful. For example, if a response has the HTTP Status code "400 Bad Request", the ***X-EM7-status-code*** might be "FAILED_INPUT_VALIDATION".
- ***X-EM7-status-message***. A human-readable description of the result of a request. The ***X-EM7-status-message*** can contain multiple messages delimited by a newline character (`\n`). For example, if a response has the HTTP Status code "302 Found", the ***X-EM7-status-message*** might be "ticket index requires a limit", indicating the request was missing the required `limit` option.
- ***X-EM7-Last-updated***. This header is returned only when requesting device configuration data from the API. Returns the date and time that at least one value in the returned data changed.

Request Headers

The following ScienceLogic-specific headers can be used when making an API request:

- ***X-em7-beautify-response***. By default, responses from the API use the minimum required amount of whitespace. If you are making requests using a tool that does not format the output (such as command line `cURL`), specify the ***X-em7-beautify-response*** header with a value of "1" to request additional whitespace in the response to make it easier to read.

CAUTION: Using the *X-em7-beautify-response:1* header can greatly increase the amount of time required to process a request. Do not use this header in integration code.

- ***X-em7-run-as***. The X-em7-run-as header can be used by administrator users to execute a request as a different user.

IMPORTANT: The value for this header must be a number that matches the account ID of an active user in your Skylar One system.

- ***X-em7-suppress-logging***. If the system is configured to write to an entry in the ticket log when a ticket is modified via the API, the *X-em7-suppress-logging* header can be used to modify a ticket via the API without updating the ticket log. If the *X-em7-suppress-logging* header with a value of "1" is included in an API request that modifies a ticket and the request is authenticated by an administrator user, the ticket logs will not be updated based on the result of the request.
- ***X-em7-send-notification***. When a ticket is created or updated, Skylar One can automatically send notification Emails to the ticket assignee and ticket watchers. If the system is not configured to send notification Emails when tickets are created or updated using the API, the *X-em7-send-notification* header can be used to send notification Emails for a specific request. If the *X-em7-send-notification* header with a value of "1" is included in an API request that modifies a ticket, notification Emails will be sent based on the result of the request.

Response Formats

The API can respond in XML and JSON formats. Use one of the following accept headers in your requests:

- ***accept: application/json, */****. The API will respond in JSON format. If the accept header is **/**, the API will respond with JSON as the default response format; however, it is recommended that you explicitly accept *application/json* for clarity.
- ***accept: application/xml, */****. The API will respond in XML format.

If the accept header for a request does not include ***application/xml***, ***application/json*** or ****/****, the API will respond with a "406 Method Not Acceptable" status code.

The contents of responses are described in the [Resources & URIs](#) section.

Chapter

3

Resources & URIs

Overview

This chapter covers the available resources for the ScienceLogic API and information about creating and updating API resources.

This chapter covers the following topics:

<i>Available Resources</i>	20
<i>URI Formatting</i>	21
<i>Resource Index Responses</i>	22
<i>Constructing URIs Using a searchspec</i>	23
<i>Filters</i>	24
<i>Required Options for Indexes</i>	27
<i>Resource Responses</i>	27
<i>Creating and Updating Resources</i>	28
<i>Asynchronous Operations</i>	29
<i>Links Between Resources</i>	29
<i>Size Limits</i>	30

Available Resources

You can interact with the following entities through the API:

- *Accounts*
- *Account Lockouts*
- *Alerts*
- *Appliances*
- *Assets*
- *Collector Groups*
- *CBQoS Objects*
- *Collection Labels*
- *Credentials*
- *Custom Attributes*
- *Dashboards*
- *Devices*
- *Device Categories*
- *Device Classes*
- *Device Interfaces*
- *Device Groups*
- *Device Relationships*
- *Device Templates*
- *Discovery Sessions*
- *Dynamic Applications*
- *Events*
- *Event Categories*
- *External Contacts*
- *File Uploads*
- *Interfaces*
- *Monitoring Policies*
- *Organizations*
- *Performance Data*
- *PowerPacks*
- *Product SKUs*
- *Schedules*

- *System Patches*
- *System Settings*
- *Tasks*
- *System Thresholds*
- *Themes*
- *Thresholds*
- *Tickets*
- *Ticket Categories*
- *Ticket Chargeback*
- *Ticket Logs*
- *Ticket Notes*
- *Ticket Queues*
- *Ticket States*
- *User Policies*
- *Vendors*

NOTE: Some resources support only view access to the corresponding Skylar One entity, while other resources provide support for create, edit, and/or delete operations. For a full listing of all actions that can be performed through the API, see the **Available Actions** appendix in the **ScienceLogic API** manual.

URI Formatting

All resources have a URI relative to the base URI for the API:

- For Database Servers, Administration Portals, and All-In-One Appliances, the base URI of the API is:

```
https://<ip-address or hostname of appliance>/api
```

The full URI for a resource has the following structure:

```
<base URI of the API><resource-uri>
```

For the resource URIs listed in the previous section, the full URI of the index is:

```
<base URI of the API>/<resource-name>
```

The URIs for specific resources combine the resource index URI and the unique ID of the specific resource. For example, the URI for the ticket with ticket ID 1 is:

```
/ticket/1
```

Some resources include sub-resources. For example, a note is a sub-resource of a ticket. If a resource includes a sub-resource, each instance of that resource includes an index for the sub-resource. For example, the index of notes attached to the ticket with ticket ID 1 is:

```
/ticket/1/note
```

And the URI for a specific note attached to ticket 1 is:

```
/ticket/1/note/<note ID>
```

NOTE: If you are accessing the API through an Administration Portal, Database Server, or All-In-One Appliance, the "/api" portion of the base URI is included in all resource URIs returned by the API.

Resource Index Responses

When you perform a GET request using the URI for a resource index, the response includes the following structure in JSON format:

```
{
  "searchspec": {
    "fields": {
      "data": [
        "field",
        .
        .
      ]
    },
    "options": {
      "option name": {
        "type": "...",

```

```

        "description": "...",
        "default": "...",
    };
    .
    .
},
},
"total_matched": "X",
"total_returned": "Y",
"result_set": [
    {
    }
]
}

```

The XML response for the same request contains the same attributes in a similar structure.

The following sections are included in the response:

- ***searchspec***. Contains filters and options that you can add to the resource index URI.
- ***total_matched***. An integer that indicates the maximum number of resources the index *could* return in the `result_set`. Resources included in this count match the requested filters but might not be included in the response because of the specified options, or because a required option is missing.
- ***total_returned***. An integer that indicates the number of resources contained in the `result_set`.
- ***result_set***. Contains each specific resource that matches the filters included in the request URI.

Constructing URIs Using a searchspec

A GET request for a resource index responds with a "searchspec" section by default. The searchspec indicates the filters and options that can be added to a resource index URI to limit or change the results contained in the response. Filters and options are added to the URI as standard GET values:

```
<resource uri>?<option 1>&<option 2>&<filter 1>&<filter 2>
```

Any number of options and filters can be added to the URI after the question mark (?), delimited by ampersands (&).

All resource indexes support an additional option that allows you to specify the sort order. The sort order option can be included only once in a single request.

Filters

You can filter the results contained in the response using any of the fields contained in the "fields" section of the searchspec. For basic equality operations, filters have the following syntax:

```
filter.<field name>=<value to equate>
```

You can add the following operators before the equals sign (=) to perform different comparisons:

- **.min**. The specified value is the minimum value for the field. Equivalent to a "greater than or equal to" operation.
- **.max**. The specified value is the maximum value for the field. Equivalent to a "less than or equal to" operation.
- **.contains**. The field contains the specified value as a sub-string.
- **.begins_with**. The field begins with the specified value as a sub-string.
- **.ends_with**. The field ends with the specified value as a sub-string.
- **.isnull**. The specified value must be 0 or 1. If you specify a value of 0, records that have a non-null value in the specified field will be returned. If you specify a value of 1, records that have a null (empty) value in the specified field will be returned.
- **.in**. The field equates to one of the values given in a list. The value to equate must be in the following list format:

```
<value 1>, <value 2>, <value 3>, ...
```

For example, to request only tickets that have a severity of major or critical (severity > 3), add the following filter clause to the ticket URI:

```
filter.severity.min=3
```

The inverse of a filter can be created by adding ".not" to the filter clause. To request the inverse of the previous example:

```
filter.severity.not.min=3
```

NOTE: If you include multiple filters for the same field in a URI, the API will return only results that match all the filters for that field (i.e. the API will perform an AND operation).

Options

Every resource index has a set of options that can be added to a request URI to limit or change the results contained in the response. Each entry in the "options" section of the searchspec has the following attributes:

- **type**. The data type of the option value. The value you pass for this option must be of this data type.
- **description**. A description of how the option affects the response.
- **default**. The default value of the option.

The following options are available on most resource indexes:

- **extended_fetch**. By default, the `result_set` will contain only the URI and description for each returned resource. If `extended_fetch` is set to 1 in the URI, the response will contain all attributes of all returned resources.
- **hide_filterinfo**. If this option is set to 1 in the URI, the response will contain only the `result_set`.
- **limit**. The maximum number of resources that should be returned in the response. For example, if you include "limit=100" in the URI, the first 100 resources are returned in the response.
- **offset**. After the API has assembled a list of possible resources to include in the response, based on the specified filters, **offset** determines which resource will be the first entry in the response list. **offset** begins at zero for the first resource, one for the second resource, and so forth. For example, if you include "limit=5&offset=5" in the `/ticket` URI, the response contains tickets six through ten from the list of the possible tickets.
- **link_disp_field**. If the **extended_fetch** option is not enabled, you can use the **link_disp_field** to specify which field will be used to populate the description for each resource. For example, the default description of each resource returned by the `/account` resource index is the username. If you want the description of each resource returned by the `/account` resource index to be the primary Email address of each user, set the **link_disp_field** option to *email*.

NOTE: Although the above options are common to most resource indexes, not all resource indexes support all of these options.

Use the following syntax to specify an option:

```
<option name>=<option value>
```

For example, to request 10 tickets with all attributes returned from the ticketing index, use the following URI:

```
/ticket?limit=10&extended_fetch=1
```

Sorting

You can sort the order of results in the response by using the **order** option. This option is available for every resource index. The syntax of the **order** option is:

```
order.<field name>=<sort order>
```

Valid values for the sort order are:

- **ASC**. Sort in ascending order.
- **DESC**. Sort in descending order
- **<value 1>,<value 2>,<value 3>,...,***. Return the records that have value 1 as the value for the field first, then the records that have value 2 as that value for the field, etc. Any number of specific values can be specified, followed by an asterisk.
- ***,<value 1>,<value 2>,<value 3>,...** Return all items that do not have one of the specified values as the value for the field, then return the records that have value 1 as the value for the field, then the records that have value 2 as that value for that field, etc. Any number of specific values can be specified.

For example, to sort the response for the /account resource by descending username, include the following option:

```
order.user=DESC
```

For example, to sort the response for the /account resource with the user accounts in organization 2 first, then all other user accounts, you would include the following option:

```
order.organization=/api/organization/2,*
```

Specifying a Query String in the Request Body

The API accepts a maximum URL size of 8 kb. If you need to perform a GET request with a query string that includes options and filters that would cause the URL to be larger than 8 kb, you can specify the query string in the body of the request. To do this:

- Do not include the query string when specifying the URL in the request
- Include the query string in the body of the request, excluding the leading question mark character
- Include the content-type header "content-type:application/x-www-form-urlencoded" in the request

For example, the following cURL request specifies a GET request to the /ticket API that includes options and filters as part of the URL:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/ticket?limit=100&extended_
fetch=1&filter.severity=3"
```

The following cURL performs the same request, but specifies the query string in the body of the request and includes the correct content-type header:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/ticket" -H "content-type:application/x-www-
form-urlencoded" -X GET -d 'limit=100&extended_fetch=1&filter.severity=3'
```

Required Options for Indexes

When you perform a GET request on some resource indexes, one or more options may be required. If a required option is missing, the response will contain a "302 Found" Status Code. The "Location:" header in the response will contain the URI for the resource with the option added. Typically the required option is a limit, which prevents responses from becoming too large.

Resource Responses

If you perform a GET request using the URI for a specific resource, the response has the following structure in JSON format:

```
{
  "field": "value",
  .
  .
  .
  "custom_fields": {
  },
  "sub_resource": {
    "URI": "...",
    "description": "...",
  },
  .
  .
```

```
}
```

The XML response for the same request contains the same attributes in a similar structure.

The following sections are included in the response:

- **field:value pairs.** In the structure shown above, *field* is the name of an attribute that is common to every resource of that type, e.g. "severity" for a ticket resource. *value* is the value of the attribute for this specific resource.
- **custom_fields.** Has the same structure as the "field": "value" pairs, but for custom fields specific to this resource type in this Skylar One systems.
- **sub resource links.** In the structure shown above, *sub resource* is name of a sub resource associated with the resource type, e.g. "notes" for a ticket resource. Each sub resource in the response contains a URI for the sub-resource index and a description of the sub resource.

Creating and Updating Resources

To modify a resource, PUT or POST XML or JSON data to the resource URI.

The XML or JSON you include in a POST or PUT request must have the same format as an XML or JSON response from a GET request on the same resource. For example, if you:

1. Perform a GET request on a ticket resource and save the response in a file.
2. In the saved file, modify the value in a single field.
3. POST the XML or JSON back to the same ticket URI.

The modified field will be updated in the ticket.

When using POST to update a resource, the XML or JSON can contain only the fields that need to be updated; any fields you want to remain the same can be removed from the XML or JSON.

To create a new resource using a POST request, you must use the URI of the resource index. The new resource will be assigned a unique ID. The API returns the URI for the new resource in the response.

In the XML or JSON structure used in a POST request, the format of the data in each field must be identical to the format the API uses when responding to GET requests. For example:

- Timestamps must be in UNIX timeticks format.
- User passwords must be an MD5 hash of the actual password.

NOTE: If you create a new resource using POST, the API ignores any links to sub-resources included in the XML or JSON structure. The response contains new URIs for sub-resource indexes.

NOTE: For information on the difference between PUT and POST, see the [HTTP Methods, Headers and Response Formats](#) section.

NOTE: If you use GraphQL for a bulk update, GraphQL will make multiple single calls to the REST API rather than one bulk call, even if Skylar One does not use the bulk capability.

Asynchronous Operations

Asynchronous operations, such as starting a discovery session, can be performed using the POST method with the "application/em7-resource-uri" content type. The "application/em7-resource-uri" content type is proprietary to the ScienceLogic API.

The following actions are performed by POSTing an em7-resource-uri to another resource:

- **Starting a discovery session.** POST a /discovery_session resource URI to the /discovery_session_active resource index.
- **Applying a device template.** POST a /device_template resource URI to a specific /device or /device_group resource.
- **Performing a "Save As" operation on a dashboard.** POST a /dashboard resource URI to the /dashboard resource index. All properties of the dashboard are copied, including those that cannot be modified directly through API requests.
- **Installing a PowerPack.** POST a /filestore/powerpack resource URI to the /powerpack resource index.
- **Registering a Patch.** POST a /filestore/system_patch resource URI to the /system_patch resource index.
- **Staging a Patch.** POST a /system_patch resource URI to the /system_patch_stage resource index.
- **Installing a Patch.** POST a /system_patch_stage resource URI to the /system_patch_deploy_active resource index.

For an example of how this content type is used, see [the Example: Using the Discovery Resource section](#).

Links Between Resources

For fields in a resource that refer to another resource, the value for the field is the URI of the other resource. For example, if you request a ticket resource that is aligned to the System organization, the "organization" field contains the URI for the resource that represents the System organization:

```
"organization": "\/organization\/0",
```

NOTE: This example shows the response from a Skylar One PowerFlow in JSON format with the forward slash characters (/) escaped. If you are accessing the API through an Administration Portal, Database Server, or All-In-One Appliance, the "/api" portion of the base URI is included in all resource URIs returned by the API.

If you are creating, updating or replacing a resource that includes links to other resources, ensure that you include the URI for the other resource in the appropriate fields.

Size Limits

The API has the following limits for URI length and POST content:

- The maximum URI length that can be used in an API request is 8199 characters.
- The maximum size of JSON content that can be included in a POST request to the API is 2 GB.
- The maximum size of XML content that can be included in a POST request to the API is 1,000,000 characters.

Chapter

4

Authentication and Access Permissions

Overview

This chapter describes the authentication and access permissions needed to use the ScienceLogic API.

This chapter covers the following topics:

<i>User Access to the API</i>	32
<i>API Keys</i>	32
<i>Account Lockouts</i>	32
<i>The _self Resource</i>	33
<i>Audit Logging</i>	34

User Access to the API

User access to the API is controlled in the same way user access to the Administration Portal is controlled:

- A user can interact only with entities associated with their organizations. Entities are either explicitly aligned with organizations, aligned with organizations based on the user that created the entity, or are not aligned with an organization.
- Users of type "Administrator" can perform all actions on all resources, regardless of organization membership.
- Device groups and dashboards can be configured so that a user must be granted a specific access key to use that device group or dashboard.

NOTE: The default user interface (AP2) architecture requires API access for all users; API access is automatically granted to users.

This chapter describes how the access permissions system applies to the API. For more information on the access permissions system in Skylar One, see the ***Access Permissions*** manual.

NOTE: User accounts that use a SAML provider for authentication cannot perform API requests unless the authentication profile for that user also includes an Internal or AD/LDAP authentication resource.

API Keys

API keys are dedicated credentials that allow scripts, integrations, and automation tools to authenticate with the Skylar One REST and GraphQL APIs without requiring a human user account. API keys are separate from personal login credentials and can be scoped to a specific user policy, making them easier to manage, audit, and revoke than traditional username-and-password authentication.

For more information about API keys, see the section on [Using API Keys in Skylar One](#).

Account Lockouts

The account lockout functionality applies to API requests (i.e., if an incorrect password is specified in multiple, sequential API requests for a valid user account, the user account will be locked out). The following settings in the **Behavior Settings** page (System > Settings > Behavior) control account lockouts:

- **Account Lockout Attempts.** Number of times a user can supply incorrect login information (i.e., the number of consecutive API requests with an incorrect password before a lockout occurs). Choices are 1 time through 10 times.
- **Account Lockout Type.** If a user enters incorrect login information multiple times in a row, that user will be locked out of the user interface. In this field, you can select how the lockout will be applied. Choices are:
 - *Lockout by IP Address.* All login attempts from the IP address will be denied.
 - *Lockout by Username and IP Address.* All login attempts by the username from the IP address will be denied.
 - *Lockout by Username (default).* All login attempts by the username will be denied.
 - *Disabled.* Lockouts are disabled.
- **Account Lockout Duration.** Specifies how long a user will be locked out of the user interface. Choices are 1 hour through 24 hours, in one-hour increments.

While a user account is locked out, API requests specifying that user will return an HTTP 403 status code with the following ScienceLogic-specific header values:

```
X-EM7-status-message: Authentication failed due to lock
```

```
X-EM7-status-code: LOCKED
```

```
X-EM7-info-message: Authentication temporarily locked due to too many
failed authentication attempts
```

Account lockouts can be removed via the API using the `/access_lock` resource. The `/access_lock` resource supports the following methods:

Action	URI	Method
View a list of locked-out user accounts.	<code>/access_lock</code>	GET
View details about a locked-out user account.	<code>/access_lock/X</code>	GET
Clear a lock on a user account.	<code>/access_lock/X</code>	DELETE

The `_self` Resource

User accounts are granted access to their own user account information through the following resource:

```
/account/_self
```

This resource returns the equivalent of the standard `/account` resource for the user that authenticated the request, even if that user account has not been granted permission to access other `/account` resources.

Audit Logging

All requests that use a PUT, POST, or DELETE method are included in the audit logs for the user's primary organization. Organizational audit logs are accessible through the **[Logs]** tab in the **Organizational Summary** page; a log for all organizations is displayed on the **Audit Logs** page (System > Monitor > Audit Logs). Each log message generated by an API request includes the following information:

1. The date when the request was made.
2. The user account that was used to authenticate the request.
3. The method used in the request.
4. The resource URI the request was made on.
5. The result of the request.

All API audit logs have a **Source** of "API Server".

Chapter

5

Using API Keys in Skylar One

Overview

API keys are dedicated credentials that allow scripts, integrations, and automation tools to authenticate with the Skylar One REST and GraphQL APIs without requiring a human user account. API keys are separate from personal login credentials and can be scoped to a specific user policy, making them easier to manage, audit, and revoke than traditional username-and-password authentication.

In Skylar One, you can use API keys to authenticate API requests. This section describes how to enable and use the feature in Skylar One.

IMPORTANT: To use this feature, you must be running Skylar One version 12.5.20 or later with AP2 9.13.1-14 (Quesito) or later installed.

This chapter covers the following topics:

<i>How API Keys Work in Skylar One</i>	36
<i>Best Practices for API Keys</i>	36
<i>Enabling API Keys in Skylar One</i>	37
<i>Enforcing a Minimum Expiration Window for API Keys</i>	37
<i>Assigning a User Policy for API Key Accounts</i>	38
<i>Managing API Keys in GraphQL</i>	38
<i>Managing API Keys in Skylar One</i>	42
<i>Using an API Key to Authenticate API Requests</i>	45

How API Keys Work in Skylar One

When an API request includes an `Authorization: Bearer <token>` header, Skylar One evaluates the token before any other authentication flow. If the token is a valid API key, Skylar One constructs the request context from the key's assigned user policy, organization membership, and permissions, all without creating a user session.

Each API key consists of two components:

- **Access Key ID.** A unique token identifier. This value is used as the bearer token in the `Authorization` header of API requests.
- **JSON Web Token (JWT).** A one-time secret token associated with the access key. The secret token cannot be retrieved after the creation modal is closed. If the secret token is lost, the key must be deleted and recreated.

API keys offer several operational advantages over basic authentication credentials:

- **Scoped permissions.** Each key is assigned a specific user policy. Revoking a key does not affect other keys, nor the user account that created it.
- **Immediate revocation.** Deleting a key instantly blocks all requests that use it. You do not need to reset any passwords.
- **Auditability.** Each key can be assigned a descriptive label (for example, "CI pipeline key" or "monitoring agent key"), making it possible to trace API activity to a specific integration or tool.
- **Reduced credential exposure.** API keys are separate from account passwords. A compromised key can be rotated without affecting the broader account.

IMPORTANT: API keys are not inherently more secure than passwords from a cryptographic standpoint; both are shared secrets transmitted in HTTP headers. The security benefit to using them comes from their granular access control, easy rotation, and per-client auditability.

Best Practices for API Keys

For optimal security, follow these guidelines when creating and managing API keys:

- **Use descriptive labels.** Assign a meaningful description to each key so you can identify its purpose in the key list and in audit logs. For example: "Automation pipeline - Production" or "ITSM integration - Staging".
- **Assign the least-privilege user policy.** Select a user policy that grants only the permissions required by the integration. Avoid assigning administrative policies to API keys unless necessary.
- **Set an expiration for short-lived integrations.** For temporary automation tasks or testing, set a key expiration rather than creating a permanent key.

- **Rotate keys regularly for long-lived integrations.** Periodically delete and recreate keys used by long-running integrations, and update the integrations with the new access key and secret.
- **Store keys securely.** Do not embed API keys directly in source code or configuration files that are stored in version control. Use a secrets manager or environment variable approach to supply keys to integrations at runtime.
- **Revoke unused keys promptly.** If an integration is decommissioned or a key may have been exposed, delete the key immediately.

Enabling API Keys in Skylar One

To use API keys in Skylar One, you must enable the feature in the `master.system_settings_general` database table. To do so:

1. Go to the console of the Database Server and run the following script:

```
silo_mysql -e "UPDATE master.system_settings_general SET value='1'
WHERE param='api_keys'"
```

2. Edit the `/opt/em7/nextui/nextui.conf` file. To do so, enter an SSH session in the console of the Database Server and run the following command at the shell prompt:

```
sudo vi /opt/em7/nextui/nextui.conf
```

3. Add the following line at the bottom of the **NextUI** configuration file:

```
INVENTORY_SHOW_API_KEYS=enabled
```

4. Save your changes, then reload the daemon service by running the following command:

```
systemctl daemon-reload
```

5. Restart the NextUI service:

```
systemctl restart nextui
```

Enforcing a Minimum Expiration Window for API Keys

Optionally, if you want to enforce a minimum expiration window on API keys, go to the console of the Database Server and run the following script to set the `api_key_min_exp` value:

```
INSERT INTO master.system_settings_general (param, value)
```

```
VALUES ('api_key_min_exp', '<minutes>')
```

```
ON DUPLICATE KEY UPDATE value = '<minutes>;'
```

Assigning a User Policy for API Key Accounts

Two default user policies that are purpose-built for API key accounts are included in Skylar One 12.5.20 and later:

- **API Read-Only User.** This policy provides read-only API access.
- **API User (No Portal Access).** This policy provides full API access.

NOTE: Both user policies prevent the user from logging in to the Skylar One user interface.

When creating an API key, assign one of these user policies to the key to ensure the key account has the right scope and cannot be used to log in to the Skylar One user interface.

Managing API Keys in GraphQL

This section describes how to create and delete API keys in GraphQL (GQL).

NOTE: Unlike in the Skylar One user interface, there is no GQL mutation for regenerating API keys. Instead, you must delete the current API key and then create a new one.

Creating an API Key in GraphQL

You can create API keys through the `createApiKey` GraphQL (GQL) mutation.

NOTE: Alternatively, if you have AP2 Quesito or later installed, you can [create an API key through the Skylar One user interface](#).

To create an API key in GQL, open the **GQL Browser** page (System > Tools > GQL Browser) and enter the following mutation:

```
mutation {
  createApiKey(
    accountPolicy: <policy_id>
    organization: <org_id>
    description: <description>
    expiration: <minutes>
  ) {
    id
    jwt
    jti
  }
}
```

where you replace:

- `<policy_id>` with the ID of the user policy you want to assign to the key. The user policy determines the permissions available to any request authenticated with this key. For more information about user policies, see the section on [User Policies](#).
- `<org_id>` with the ID of the primary organization you want to associate with the key. This determines the organizational context in which the key operates.
- `<description>` with a descriptive label for the key, such as "Monitoring agent" or "CI pipeline integration". This label appears in the API keys list and in audit logs.
- `<minutes>` with the number of minutes after which the key will expire.

TIP: See the [Best Practices for API Keys](#) section for suggested guidelines relating to user policies, descriptions, and expirations.

The response returns the following information:

- `id`. The unique numeric ID assigned to the API key by Skylar One.
- `jwt`. The JSON Web Token (JWT), which is a one-time secret token associated with the access key.
- `jti`. The JWT ID, which is a unique token identifier used as the bearer token in the `Authorization` header of API requests.

IMPORTANT: Copy and store both the JWT token and the JWT ID immediately. The secret token cannot be retrieved after this modal is closed. If the token is lost, you must delete the key and create a new one.

Deleting an API Key in GraphQL

You can delete API keys through the `deleteApiKey` GraphQL (GQL) mutation. When you do so, the JSON Web Token (JWT) for the API key is revoked immediately.

NOTE: Alternatively, if you have AP2 Quesito or later installed, you can [delete an API key through the Skylar One user interface](#).

To delete an API key in GQL, open the **GQL Browser** page (System > Tools > GQL Browser) and enter the following mutation, replacing `<key_id>` with the unique numeric ID that Skylar One assigned to the API key:

```
mutation {
  deleteApiKeys (
    ids: <key_id>
  ) {
    successes,
    failures,
    invalid
  }
}
```


Querying API Keys

NOTE: Because it appears only once at the time of its creation, you cannot view the JSON Web Token (JWT) for an API key using a GQL query.

To get information about an API key in GraphQL (GQL), open the **GQL Browser** page (System > Tools > GQL Browser) and enter the following query, replacing `<key_id>` with the unique numeric ID that Skylar One assigned to the API key:

```
query get {
  apiKey(id: "<key_id>") {
    id
    jti
    description
    expiration
    createdOn
    lastLogin
    organization {
      id
      company
      city
    }
    additionalOrganizations {
      id
      company
    }
    accountPolicy {
      id
      name
    }
  }
}
```

The query returns the following information:

- `id`. The unique numeric ID assigned to the API key by Skylar One.
- `jti`. The JWT ID, which is a unique token identifier used as the bearer token in the `Authorization` header of API requests.
- `description`. A label assigned to the key at creation time.
- `expiration`. The date and time the key expires.

- `createdOn`. The date and time the key was created.
- `lastLogin`. The date and time the key was last used to authenticate a request.
- `organization`. The primary organization associated with the key.
- `additionalOrganizations`. Additional organizations associated with the key.
- `accountPolicy`. The user policy assigned to the key, which determines the key's permissions.

Managing API Keys in Skylar One

This section describes how to create, view, and delete API keys in the user interface.

NOTE: The **API Keys** page (Manage > API Keys), which enables you to view, create, and delete API keys in the Skylar One user interface, is available only if you have installed AP2 9.13.1-14 (Quesito) or later and enabled the feature on your Skylar One system. If the **API Keys** page does not appear in your navigation, contact your ScienceLogic administrator.

To access the **API Keys** page, you must have an Administrator user account. For more information on access hooks, see the section on [Access Permissions](#).

Viewing the List of API Keys

From the **API Keys** page (Manage > API Keys), you can view all API keys that are visible to your user account.

The **API Keys** page displays the following information about each key:

- **API Key Name.** The key's JSON Web Token (JWT) ID.
- **Description.** A label assigned to the key at creation time.
- **API Key ID** The unique numeric ID assigned to the API key by Skylar One.
- **Expiration.** The date and time the key expires.
- **Primary Organization.** The organization or organizations associated with the key.
- **Aligned User Policy.** The user policy assigned to the key, which determines the key's permissions.
- **Created On.** The date and time the key was created.

TIP: To sort the list, click on a column heading. The list will be sorted by the column value, in ascending order. To sort the list by descending order, click the column heading again. You can also filter the items on this inventory page by typing filter text or selecting filter options in one or more of the filters found above the columns on the page. For more information, see [Filtering Inventory Pages](#) in the *Introduction to Skylar One* manual.

TIP: You can use the **Search** field to look for specific items in the list. For more information, see [Using Basic Search](#) in the *Introduction to Skylar One* manual. If you are looking for a very specific set of device classes, click the gear icon (⚙️) to the right of the **Search** field and select *Advanced*. In this mode, you can create an advanced search using "AND" or "OR" for multiple search criteria. For more information, see [Performing an Advanced Search](#) in the *Introduction to Skylar One* manual.

TIP: If you do not see one of these columns, click the gear icon (⚙️) and then select *Column Preferences* to add or remove columns. You can also drag columns to different locations on the page or click on a column heading to sort the list by the values in that column. Skylar One retains any changes you make to the columns that appear on the page and will automatically recall those changes the next time you visit the page. For more information, see the section on [Editing the Settings for an Inventory Page](#) in the *Introduction to Skylar One* manual.

Creating an API Key in the Skylar One User Interface

To create a new API key:

1. Go to the **API Keys** page (Manage > API Keys).
2. Click the **[Create New]** button. The **Create API Key** modal appears.
3. In the **Create API Key** modal, enter values in the following fields:
 - **Description.** Optionally, enter a descriptive label for the key, such as "Monitoring agent" or "CI pipeline integration". This label appears in the API Keys list and in audit logs.
 - **Expiration.** Select an expiration date and time for the key. The default value is 1440 minutes (24 hours).
 - **Organization.** Select the primary organization to associate with the key. This determines the organizational context in which the key operates.
 - **Additional Organizations.** Optionally, select one or more additional organizations to associate with the key.
 - **Account Policy.** Select the user policy to assign to the key. The user policy determines the permissions available to any request authenticated with this key. For more information about user policies, see the section on [User Policies](#).

TIP: See the [Best Practices for API Keys](#) section for suggested guidelines relating to user policies, descriptions, and expirations.

4. Click **[Next]**. Confirm your selections and then click **[Create API Key]**.

5. The **API Key Created** modal appears and displays the **Key ID** and **JWT Token** for the API key.

IMPORTANT: Copy and store both the key ID and the JWT token immediately. The secret cannot be retrieved after this modal is closed. If the token is lost, you must delete the key and create a new one.

6. Click **[Done]** to dismiss the modal.

Viewing API Key Details

To view details about an existing API key:

1. Go to the **API Keys** page (Manage > API Keys).
2. Click the **API Key Name** for the key you want to view, or click the **Actions** icon (⋮) and select *Details*. The **View API Key Details** modal appears.

The modal displays the following:

- **API Key Name.** The key's JSON Web Token (JWT) ID.
- **API Key Description.** The label assigned to the key at creation time.
- **Expiration.** The expiration date and time.
- **User Policy Name.** The user policy assigned to the key, which determines the key's permissions.
- **API Key ID.** The unique numeric ID assigned to the API key by Skylar One.
- **Created On.** The date and time the key was created.
- **Last Login.** The date and time the key was last used to log in to the ScienceLogic API.
- **Primary Organization.** The primary organization associated with the key.
- **Aligned Organizations.** All of the organizations associated with the key.

NOTE: API keys cannot be edited in place. To modify a key's policy, organization, or other settings, you must delete the existing key and create a new one.

Regenerating an API Key

When you regenerate an API key, it deletes the current API key and creates a new one with the same user policy, organization alignments, and description. In doing so, the previous API key also stops working.

To regenerate an API key:

1. Go to the **API Keys** page (Manage > API Keys).
2. Click the **Actions** icon (⋮) for the key you want to delete and select *Regenerate*.
3. On the **Confirm API Key Regeneration** modal, click **[Confirm]**.

4. The **API Key Regenerated** modal appears and displays the **API Key ID**, **API Key Name**, and **JWT Token** for the API key.

IMPORTANT: Copy and store both the key name and the JWT token immediately. The secret cannot be retrieved after this modal is closed. If the token is lost, you must delete the key and create a new one.

5. Click **[Close]** to dismiss the modal.

Deleting an API Key

Deleting an API key immediately and permanently revokes it. Any requests that use the deleted key will receive a "401 Unauthorized" response.

To delete an API key:

1. Go to the **API Keys** page (Manage > API Keys).
2. Click the **Actions** icon (⋮) for the key you want to delete and select *Delete*.
3. In the confirmation dialog, click **[Delete]** to confirm.

The key is removed from Skylar One. All authentication attempts using that key will be rejected immediately.

NOTE: You cannot undo the deletion of an API key. If you delete a key that is in use by an integration, that integration will stop functioning until it is updated with a new key.

TIP: To delete multiple keys on the **API Keys** page (Manage > API Keys), select the checkboxes of the keys you want to delete and then click the **[Delete API Keys]** button.

Cleaning Up Expired Keys

Skylar One automatically removes expired API keys. When a key's expiration date and time passes, Skylar One deletes the key's account record and associated metadata.

Expired keys are not valid for authentication and do not appear on the **API Keys** page (Manage > API Keys) after they are removed.

Using an API Key to Authenticate API Requests

To authenticate a REST or GraphQL API request using an API key, include the access key as a bearer token in the `Authorization` header of the request:

```
Authorization: Bearer <access_key>
```

Skylar One validates the token and, if valid, processes the request under the permissions defined by the key's assigned user policy and organization membership.

Example REST request:

```
GET https://<skylar_one_host>/api/v1/devices
Authorization: Bearer <access_key>
```

Example GraphQL request:

```
POST https://<skylar_one_host>/graphql
Authorization: Bearer <access_key>
Content-Type: application/json

{ "query": "{ devices { id name } }" }
```

NOTE: API keys cannot be used to log in to the Skylar One user interface. Attempting to use an API key's access key and secret on the Skylar One login page will return a generic authentication error.

Chapter

6

Custom Attributes

Overview

This chapter describes how to view, add, and edit custom attributes for API resources.

This chapter covers the following topics:

<i>Custom Attributes for API Resources</i>	48
<i>Viewing and Adding Custom Attributes</i>	48
<i>Example of How to Add Custom Attributes</i>	51
<i>Editing Custom Attributes</i>	53
<i>Requests to Resources with Custom Attributes</i>	54
<i>Removing Custom Attributes</i>	56

Custom Attributes for API Resources

The ScienceLogic API includes resources for adding custom attributes to the following resources:

- /asset
- /device
- The /interface sub-resource under /device resources
- /theme
- /vendor

When you define a custom attribute for a resource:

- For any instance of that resource (e.g., a specific device), you can perform a POST operation specifying a value for that attribute for that instance.
- If you configure the attribute as a base attribute, the attribute will appear in the list of fields for all instances of that resource. For example, if you define a custom attribute as a base attribute for the /device resource, the response to a GET request for any /device/device_id resource includes the custom attribute in the list of fields.
- If you configure the attribute as an extended attribute, the attribute will appear in the list of fields for instances of that resource only if a value has been specified for the attribute for that instance. For example, suppose you define a custom attribute as an extended attribute for the /device resource. The response to a GET request on the /device resource index with the extended_fetch option enabled will include the custom attribute only for devices that have a value for that custom attribute.
- GET requests for the resource index can include filter and sort criteria that use that custom attribute.

When you define a value for a custom attribute by performing a POST request to a resource, the value is available through the API and can be used in dynamic rules for device groups and viewed in the custom table widget.

Viewing and Adding Custom Attributes

You can view information about the custom attributes for a resource by performing a GET request to one of the following resource indexes:

- /custom_attribute/asset
- /custom_attribute/device
- /custom_attribute/interface
- /custom_attribute/theme
- /custom_attribute/vendor
- /custom_attribute/_lookup. Allows for searching across all custom attributes for all entity types.

NOTE: The "limit" option is required for all resource indexes for custom attributes.

Each resource custom attribute resource index returns a list of custom attributes including the URI for each custom attribute. URIs for custom attributes are in the following format:

```
/custom_attribute/<resource type>/<attribute name>
```

By default, no custom attributes are defined for any of the resources that support custom attributes.

To add a custom attribute for a resource, perform a POST request to either of the following URIs:

- The corresponding `/custom_attribute/resource` resource index.
- The URI of the custom attribute itself, i.e. `/custom_attribute/resource/name`.

The body of a POST request to an `/custom_attribute/resource` resource index must have the following JSON structure:

```
{  
  "name": "attribute name",  
  "label": "attribute label",  
  "type": "attribute type",  
  "index": "attribute index type",  
  "extended": "attribute extended option"  
}
```

Or the following XML structure:

```
<custom_attribute>  
  <name>attribute name</name>  
  <label>attribute label</label>  
  <type>attribute type</type>  
  <index>attribute index type</index>  
  <extended>attribute extended option</extended>  
</custom_attribute>
```

The body of a POST request to an `/custom_attribute/resource/name` resource must have the following JSON structure:

```
{  
  
  "label": "attribute label",  
  
  "type": "attribute type",  
  
  "index": "attribute index type",  
  
  "extended": "attribute extended option"  
  
}
```

Or the following XML structure:

```
<custom_attribute>  
  
  <label>attribute label</label>  
  
  <type>attribute type</type>  
  
  <index>attribute index type</index>  
  
  <extended>attribute extended option</extended>  
  
</custom_attribute>
```

NOTE: You can request example JSON or XML content that must be posted to a `/custom_attribute/resource/name` resource by performing a GET request to the following URI: `/custom_attribute/resource/_example`.

Where *attribute name*, *attribute label*, *attribute type*, *attribute index type*, and *attribute extended option* are properties of the custom attribute you want to add. Attributes have the following properties:

- **name.** The name of the custom attribute. Names for custom attributes must conform to XML naming standards. The attribute name can contain any combination of alphanumeric characters, a period, a dash, a combining character or an extending character. If you attempt to create a custom attribute with a non-compliant name, the API will respond with a HTTP 400 Bad Request status.
- **label.** A human-readable description of the attribute, up to 128 characters in length.
- **type.** The data type of the custom attribute. You must specify one of the following two values in the **type** field:
 - *integer.* The custom attribute will be used to store signed 64-bit integer values.
 - *string.* The custom attribute will be used to store string values up to 512 characters in length.

- **index.** You must specify one of the following three values in the **index** field:
 - *index.* When Skylar One creates the database table that stores this custom attribute, the column that stores this value will be set as an index for the table. Setting index values can speed up queries performed on the database table, but does not affect which filter or search options will be available for this custom attribute.
 - *unique.* When Skylar One creates the database table that stores this custom attribute, the column that stores this value will be set as a unique index for the table. The values defined for this custom attribute must be unique for all resources. For example, if you add a custom attribute called "c-external-id" to the /custom_attribute/device resource and define the **index** as *unique*, the value of "c-external-id" for a /device/device_id resource cannot be re-used for another /device/device_id resource. Setting index values can speed up queries performed on the database table, but does not affect which filter or search options will be available for this custom attribute.
 - *none.* When Skylar One creates the database table that stores this custom attribute, the column that stores this value will not be set as an index or unique index.
- **extended.** A boolean value. You must specify 0 or 1 in this field:
 - *0.* The attribute is a "base" attribute. The attribute is displayed in the list of fields for all instances of the specified resource regardless of whether a value has been specified for the attribute.
 - *1.* The attribute is an "extended" attribute. The attribute is displayed in the list of fields for an instance of the specified resource only if a value has been specified for the attribute.

When you add a custom attribute, the default value for all resources where that attribute is now defined is NULL.

Example of How to Add Custom Attributes

Suppose you are integrating Skylar One with an external provisioning system and you want to include information from the external provisioning system in each device record to make searching for devices and generating reports easier. You could define the following two custom attributes:

- An ID value from the external provisioning system
- A name field from the external provisioning system

To add these custom attributes, you would perform two POST requests with the following JSON structures to the /custom_attribute/device resource to create the two custom attributes:

Request 1:

```
{
  "name": "external-id",
  "label": "ID from external provisioning system",
  "type": "integer",
```

```

    "index": "unique",

    "extended": "0"
}

```

Request 2:

```

{
    "name": "external-name",

    "label": "Name from external provisioning system",

    "type": "string",

    "index": "none",

    "extended": "0"
}

```

Each request specifies a custom attributes:

- ***external_id***. An integer value that will contain the ID value from the external provisioning system. The *index* field is set to *unique* because all ID values from the external provisioning system will be unique.
- ***external_name***. A string value that will contain the name from the external provisioning system.

The structures look like this in XML format:

Request 1:

```

<custom_attribute>

    <name>external-id</name>

    <label>ID from external provisioning system</label>

    <type>integer</type>

    <index>unique</index>

    <extended>0</extended>

</custom_attribute>

```

Request 2:

```

<custom_attribute>

```

```
<name>external-name</name>

<label>Name from external provisioning system</label>

<type>string</type>

<index>none</index>

<extended>0</extended>

</custom_attribute>
```

Editing Custom Attributes

To edit a custom attribute, perform a POST request to the URI for that attribute. URIs for custom attributes are in the following format:

```
/custom_attribute/<resource type>/<attribute name>
```

The body of a POST request to a `/custom_attribute/<resource type>/<attribute name>` resource must have the following JSON structure:

```
{
  "label": "attribute label",
  "type": "attribute type",
  "index": "attribute index type"
}
```

Or the following XML structure:

```
<custom_attribute>

  <label>attribute label</label>

  <type>attribute type</type>

  <index>attribute index type</index>

</custom_attribute>
```

NOTE: You cannot update the name or the extended option of a custom attribute.

Requests to Resources with Custom Attributes

When you define a custom attribute for a resource:

- If the attribute is a "base" attribute, the attribute is displayed in the list of fields for all instances of the specified resource regardless of whether a value has been specified for the attribute.
- If the attribute is an "extended" attribute, the attribute is displayed in the list of fields for an instance of the specified resource only if a value has been specified for the attribute.

NOTE: To view or define custom attributes, you must prefix the attribute key with `c-`.

For example, if you created the "external_id" and "external_name" attributes described in the [Example of How to Add Custom Attributes](#) section, both of which are base attributes, the response to a GET request for a `/device/device_id` resource would look like this:

```
{
  "name": "em7_ap",
  "ip": "10.0.9.50",
  "snmp_cred_id": "\\credential\\snmp\\1",
  "snmp_w_cred_id": null,
  "class_type": "\\device_class\\20036",
  "organization": "\\organization\\0",
  "auto_update": "1",
  "event_suppress_mask": "00:10:00",
  "auto_clear": "1",
  "log_all": "1",
  "daily_port_scan": "1",
  "critical_ping": "0",
  "scan_all_ips": "0",
  "preserve_hostname": "1",
```

```

"disable_asset_update":"0",

"date_added":"1320183224",

"c-external-id":"",

"c-external-name":"",

"parent_device":null,

"child_devices":{

},

"state":0,

"notes":{

  "URI":"/device/2/note/?hide_filterinfo=1&limit=1000",

  "description":"Notes"

},

.

.

"app_credentials":{

  "URI":"/device/2/device_app_credentials",

  "description":"Read-only lookup for aligned credentials and the
device-aligned apps that are using them"

}

}

```

To define a value for a custom attribute for a specific instance of a resource, you can include the custom attribute when performing a POST request to that resource. For example, to define a value for the "external-id" attribute for the device with ID "3", you would POST to following JSON to the /device/3 resource:

```

{

  "ip":"10.0.9.50",

```

```
"c-external-id": "4"  
}
```

When you perform a GET request on a resource index, you can use custom attributes in filter and sort criteria. For example, if you want to perform a GET request on the /device resource index and want to sort the response by the external-id field, you would request the following URI:

```
/device?limit=100&order.c-external-id=ASC
```

If you want to perform a GET request on the /device resource index and want to filter the response to include only devices that contain the string "server" in the "external-name" field, you would request the following URI:

```
/device?limit=100&filter.c-external-name.contains=server
```

Removing Custom Attributes

To remove a custom attribute, perform a DELETE request to the URI for that attribute. URIs for custom attributes are in the following format:

```
/custom_attribute/<resource type>/<attribute name>
```

NOTE: If you want to unalign a custom attribute for an interface, you can perform a PUT action and set the value to *null* (lowercase without quotes).

Chapter

7

Generating Events Using the API

Overview

The /alert API resource can be used to generate alerts in Skylar One that will appear as log messages in the **Device Logs & Messages** page, similar to how Skylar One processes inbound syslog and trap messages. You can optionally create one or more event policies that will trigger when an alert generated through the API meets the criteria specified in the policy.

This chapter covers the following topics:

<i>Generating Alerts</i>	58
<i>Defining API Event Policies</i>	59

Generating Alerts

To generate an alert, you must perform a POST request to the /alert resource index. The content you POST must have the following structure:

```
{  
  "force_ytype": "0",  
  "force_yid": "0",  
  "force_yname": "",  
  "message": "",  
  "value": "0",  
  "threshold": "0",  
  "message_time": "0",  
  "aligned_resource": ""  
}
```

Supply the following values in each field:

- **force_ytype**. Optional. The type of sub-entity on a device that you want to associate the alert with. This field can be set to the following numeric values that represent sub-entity types:
 - 1. CPU
 - 2. Disk
 - 3. File System
 - 4. Memory
 - 5. Swap
 - 6. Hardware Component
 - 7. Interface
 - 9. Process
 - 10. Port
 - 11. Windows Service

- 12. Web Content
- 13. Email Monitor

For example, to associate the alert with a specific interface on a device, supply "7" in this field. If you are not supplying information about a sub-entity, supply 0 (zero) in this field.

- **force_yid.** Optional. The ID value of the specific sub-entity on the device that you want to associate the alert with. For example, if you are associating the alert with the interface with ID 2, supply "2" in this field. If you are not supplying information about a sub-entity, supply 0 (zero) in this field.
- **force_yname.** Optional. The name of the specific sub-entity on the device that you want to associate the alert with. For example, if you are associating the alert with the interface called "eth0", supply "eth0" in this field. If you are not supplying information about a sub-entity, supply an empty string in this field.

NOTE: If an event policy is configured to clear another event policy, an instance of the event is cleared only when the clearing event has a matching sub-entity type, sub-entity ID, and sub-entity name.

- **message.** Enter message text to associate with the alert. If the alert does not match an event, this text will be displayed in the **Device Logs & Messages** page. This text will be used to match against the First Match String and Second Match String values in event policies. If the alert triggers an event, this text will be substituted for the %M substitution character in the event message.
- **value.** Optionally, supply the numeric value that triggered the alert. For example, if an alert indicates that CPU usage is high, you might pass the current CPU usage in this field. If you are not supplying a specific value, supply 0 (zero) in this field.
- **threshold.** Optionally, supply the numeric threshold that was exceeded for this alert to be generated. This threshold can be used in an event policy message by using the %T substitution. If you are not supplying a specific threshold, supply 0 (zero) in this field.
- **message_time.** The timestamp to associate with the alert in unix time format. The device log message will be listed at this date and time. Valid values include a timestamp or an empty string, "0" (zero), or "now", the latter three of which default to the current timestamp.
- When creating a new API alert, the /api/alert endpoint now allows a custom timestamp. Valid values for message_time include a timestamp or an empty string, 0, or now, the latter three of which default to the current timestamp.
- **aligned_resource.** The relative URI of the device with which you want to associate the alert. For example, to align the alert with device ID 1, supply /device/1.

Defining API Event Policies

All alerts generated using the /alert resources are matched against event policies of type "API".

When you create API event policies, the event messages are generated by inserting messages into the main database. These messages can be inserted by a snippet automation action, a snippet Dynamic Application, or by a request to the ScienceLogic API.

To define an API event policy:

1. Go to **Event Policies** page (Events > Event Policies, or Registry > Events > Event Manager in the classic user interface).
2. In the **Event Policies** page, click the **[Create Event Policy]** button. The **[Policy Description]** tab of the **Event Policy Editor** appears.
3. On the **[Policy Description]** tab, enter the following information:
 - **Policy Name.** Type a name for the event policy.
 - **Enable Event Policy.** Turn this toggle on to enable the event policy, or toggle it off to disable the event policy.
 - **Policy Description.** Type a description of the event policy.
4. Click the **[Match Logic]** tab, then enter the following information:
 - **Event Source.** Specifies the source for the event. Select *API*.
5. After selecting and defining your **Event Source**, enter values in the fields on the right side of the **Match Logic** tab:
 - **String/Regular Expression.** Use this drop-down to select *String* or *Regular Expression*.
 - **Match String.** Type a text string or a regular expression to match against the originating log message field of each alert generated through the API. The event will be generated if the message matches the **Match String** and the optional **Second Match String** values. This string can be up to 512 characters and length and can be any combination of alpha-numeric and multi-byte characters.

CAUTION: If you do not supply a value in the **Match String** field, your event policy will match all alerts generated through the API.

CAUTION: Skylar One's expression matching is case-sensitive.

- **Second Match String (Optional).** Optionally, a second text string or regular expression to match against the originating log message field of each alert generated through the API. The event will be generated if the message matches the **Match String** and the **Second Match String** values.

NOTE: The other fields on this page can be used to define specific event behavior or enable advanced event features. For a description of every option on this page, see the **Events** manual.

6. Click the **[Event Message]** tab, then enter the following information:
 - **Event Message.** Define the message that appears in the **Event Console** page or the **Viewing Events** page when this event occurs.

NOTE: For more information about the *Event Message* field and descriptions of the other fields on this page that can be used to define the event severity, event masking, and other options, see the *Events* manual.

7. Optionally, you can click the **[Suppression]** tab, where you can define specific devices or device groups for which the event should not appear.

NOTE: For more information about the **[Suppression]** tab and the fields that appear on this page, see the *Events* manual.

8. After entering information in each tab, click **[Save]** to save your new event policy.

Defining API Event Policies in the Classic Skylar One User Interface

All alerts generated using the /alert resources are matched against event policies of type "API".

To create an event policy of type "API" in the classic Skylar One user interface:

1. Go to the **Event Policy Manager** page (Registry > Events > Event Manager).
2. Click the **[Create]** button. The **Event Policy Editor** page is displayed.
3. Supply values in the following fields:
 - **Event Source.** Select API.
 - **Operational State.** Select whether the event policy is enabled or disabled.
 - **Policy Name.** Enter a name for your event policy.
 - **Event Message.** Enter the event message that will be displayed in the event console when this event is generated. You can use the %M (message), %V (value), and %T (threshold) substitution characters in this field to include information from the API request.
 - **Policy Description.** Enter descriptive text about your event policy. This text is displayed when a user selects the information icon (📘) for an instance of this event.

NOTE: The *Use Modifier* checkbox is not applicable to API event policies.

4. Click the **[Advanced]** tab. The advanced options are displayed.
5. Supply values in the following fields:
 - **First Match String.** Enter text or a regular expression to match against the message field of each alert generated through the API. The event will be generated if the message matches the *First Match String* and the *Second Match String* values.

CAUTION: If you do not supply a value in the *First Match String* field, your event policy will match all alerts generated through the API.

- **Second Match String.** Optionally, a second text string or regular expression to match against the message field of each alert generated through the API. The event will be generated if the message matches the *First Match String* and the *Second Match String* values.
- **Match Logic.** Specifies whether the *First Match String* and *Second Match String* values are matched as text strings or regular expressions.

NOTE: The other fields on this page can be used to define specific event behavior or enable advanced event features. For a description of every option on this page, see the *Events* manual.

6. Click the **[Save]** button.

Chapter

8

Requesting Performance Data in Bulk

Overview

The resources */data_performance*, */data_performance_raw*, and their sub-resources can be used to request performance data for multiple devices or interfaces in a single request. This chapter describes how to use these resources to request performance data.

This chapter covers the following topics:

<i>Resource URIs</i>	64
<i>Specifying the Time Range for a Data Request</i>	66
<i>Specifying Data Fields</i>	66
<i>Requesting Data for Specific Devices or Interfaces</i>	90
<i>Additional Options</i>	94
<i>Responses from Bulk Performance Data Resources</i>	95

Resource URIs

The following table lists the resource URIs for the resources: /data_performance and /data_performance_raw.

NOTE: For resources that return data, you must specify a timestamp option. If you do not [specify a timespan](#), the API will return an HTTP 400 (Bad Request) status code.

URI	Description
/data_performance	Returns a list of URIs for the sub-resources associated with each available entity type (device and interface).
/data_performance/device	Returns a list of URIs that can be used to request device performance data.
/data_performance/device/dynamic_app	Returns normalized (rolled-up) performance data from one or more Dynamic Applications. The data matches specified parameters.
/data_performance/device/monitor_port	Returns normalized (rolled-up) data from port monitoring policies. The data matches specified parameters.
/data_performance/device/monitor_cv	Returns normalized (rolled-up) data from web content monitoring policies. The data matches specified parameters.
/data_performance/device/monitor_tv	Returns normalized (rolled-up) data from SOAP/XML transaction monitoring policies. The data matches specified parameters.
/data_performance/device/monitor_process	Returns normalized (rolled-up) data from system process monitoring policies. The data matches specified parameters.
/data_performance/device/monitor_service	Returns normalized (rolled-up) data from Windows service monitoring policies. The data matches specified parameters.
/data_performance/device/monitor_email	Returns normalized (rolled-up) data from email round-trip monitoring policies. The data matches specified parameters.
/data_performance/device/monitor_dns	Returns normalized (rolled-up) data from DNS monitoring policies. The data matches specified parameters.
/data_performance/device/filesystem	Returns normalized (rolled-up) data from file system usage policies. The data matches specified parameters.

URI	Description
/data_performance/device/avail	Returns normalized (rolled-up) data about availability and latency. The data matches specified parameters.
/data_performance/interface	Returns normalized (rolled-up) data about interface utilization. The data matches specified parameters.
/data_performance/cbqos	Returns normalized (rolled-up) data for CBQoS metrics. The data matches specified parameters.
/data_performance_raw	Returns a list of URIs for the sub-resources associated with each available entity type (device and interface).
/data_performance_raw/device	Returns a list of URIs that can be used to request raw performance data for a device.
/data_performance_raw/device/dynamic_app	Returns raw performance data from one or more Dynamic Applications. The data matches specified parameters.
/data_performance_raw/device/monitor_port	Returns raw data from port monitoring policies. The data matches specified parameters.
/data_performance_raw/device/monitor_cv	Returns raw data from web content monitoring policies. The data matches specified parameters.
/data_performance_raw/device/monitor_tv	Returns raw data from SOAP/XML transaction monitoring policies. The data matches specified parameters.
/data_performance_raw/device/monitor_process	Returns raw data from system process monitoring policies. The data matches specified parameters.
/data_performance_raw/device/monitor_service	Returns raw data from Windows service monitoring policies. The data matches specified parameters.
/data_performance_raw/device/monitor_email	Returns raw data from email round-trip monitoring policies. The data matches specified parameters.
/data_performance_raw/device/monitor_dns	Returns raw data from DNS monitoring policies. The data matches specified parameters.
/data_performance_raw/device/filesystem	Returns raw data about file system usage. The data matches specified parameters.
/data_performance_raw/device/avail	Returns raw data about availability and latency. The data matches specified parameters.
/data_performance_raw/interface	Returns raw data about interface utilization. The data matches specified parameters.
/data_performance_raw/cbqos	Returns raw data for CBQoS metrics. The data matches specified parameters.

Specifying the Time Range for a Data Request

All requests to sub-resources of `/data_performance` and `/data_performance_raw` that return performance data must specify a time range for the returned data. If you do not specify a time range, the API will return an HTTP 400 (Bad Request) status code.

You can use the following options in the resource URI to specify a time range:

- ***duration***. Specifies the duration of the time range in human-readable shorthand format. A valid value for this option includes an integer and one of the following characters:
 - *m*. The integer specifies the number of minutes in the time range.
 - *h*. The integer specifies the number of hours in the time range.
 - *d*. The integer specifies the number of days in the time range.
- ***beginstamp***. The UNIX timestamp for the start of the time range.
- ***endstamp***. The UNIX timestamp for the end of the time range.

You must use one of the following combinations of these options:

- Specify a ***beginstamp*** and ***endstamp***. The time range starts at the time specified in the ***beginstamp*** option and ends at the time specified in the ***endstamp*** option.
- Specify a ***beginstamp*** and ***duration***. The time range starts at the time specified in the ***beginstamp*** option and covers the amount of time specified in the ***duration*** option.
- Specify a ***endstamp*** and ***duration***. The time range covers the amount of time specified in the ***duration*** option ending at the time specified in the ***endstamp*** option.
- Specify only the ***duration*** option. This is equivalent to specifying an ***endstamp*** value of the current time with the specified ***duration*** option.

For the sub-resources of `/data_performance`, you must also specify a value in the ***rollup_freq*** option. Valid values for this option are:

- *hourly*. The response will include hourly rollup data.
- *daily*. The response will include daily rollup data.

Specifying Data Fields

If you do not specify a set of data fields in your request, no data will be returned in the response.

To specify data fields, supply a comma-delimited list of fields in the `data_fields` option. The available data fields are different for each resource. The available fields for each resource are listed in the options section of the searchspec returned by the resource.

Fields for Dynamic Application Resources

For the resources */data_performance/device/dynamic_app* and */data_performance_raw/device/dynamic_app*, the *data_fields* option can include the following fields:

Field	Description
A presentation object ID. Presentation object IDs are different for each Skylar One system and can be looked up using the <i>/dynamic_app</i> resource and sub-resources.	The presentation objects for which data sets will be returned.
A presentation object GUID. Presentation object GUIDs are the same for all Skylar One system and can be looked up using the <i>/dynamic_app</i> resource and sub-resources.	The presentation objects for which data sets will be returned.

Fields for Port Monitor Resources

For the resource */data_performance/device/monitor_port*, the *data_fields* option can include the following fields:

Field	Description
avg_d_state	The average availability of the port, calculated from the raw data points for the rollup period. Availability values are either zero (0, unavailable) or one (1, available); average values will range from zero to one.
max_d_state	The value of the single highest availability poll for the port during the rollup period. Values are either zero (0, unavailable) or one (1, available).
min_d_state	The value of the single lowest availability poll for the port during the rollup period. Values are either zero (0, unavailable) or one (1, available).
sum_d_state	The sum of all availability values for the port during the rollup period. Availability values are either zero (0, unavailable) or one (1, available).
std_d_state	The standard deviation of availability values for the port, calculated from the raw data points for the rollup period.

For the resource */data_performance_raw/device/monitor_port*, the *data_fields* option can include the following fields :

Field	Description
d_state	The availability of the port. Availability values are either zero (0, unavailable) or one (1, available).

Fields for Web Content Monitor Resources

For the resource `/data_performance/device/monitor_cv`, the `data_fields` option can include the following fields:

Field	Description
<code>min_d_conn_time</code>	The lowest connection time, in seconds, of all polls during the rollup period.
<code>max_d_conn_time</code>	The highest connection time, in seconds, of all polls during the rollup period.
<code>avg_d_conn_time</code>	The average connection time, in seconds, calculated from the raw data points for the rollup period.
<code>sum_d_conn_time</code>	The sum of all connection times, in seconds, during the rollup period.
<code>std_d_conn_time</code>	The standard deviation of the connection times, calculated from the raw data points for the rollup period.
<code>min_d_dl_size</code>	The lowest download size, in bytes, of all polls during the rollup period.
<code>max_d_dl_size</code>	The highest download size, in bytes, of all polls during the rollup period.
<code>avg_d_dl_size</code>	The average download size, in bytes, calculated from the raw data points for the rollup period.
<code>sum_d_dl_size</code>	The sum of all download sizes, in bytes, during the rollup period.
<code>std_d_dl_size</code>	The standard deviation of the download sizes, calculated from the raw data points for the rollup period.
<code>min_d_dl_speed</code>	The lowest download speed, in bytes/second, of all polls during the rollup period.
<code>max_d_dl_speed</code>	The highest download speed, in bytes/second, of all polls during the rollup period.
<code>avg_d_dl_speed</code>	The average download speed, in bytes/second, calculated from the raw data points for the rollup period.
<code>sum_d_dl_speed</code>	The sum of all download speeds, in bytes/second, during the rollup period.
<code>std_d_dl_speed</code>	The standard deviation of the download speeds, calculated from the raw data points for the rollup period.
<code>min_d_ns_time</code>	The lowest DNS lookup time, in seconds, of all polls during the rollup period.
<code>max_d_ns_time</code>	The highest DNS lookup time, in seconds, of all polls during the rollup period.
<code>avg_d_ns_time</code>	The average DNS lookup time, in seconds, calculated from the raw data points for the rollup period.
<code>sum_d_ns_time</code>	The sum of all DNS lookup times, in seconds, during the rollup period.
<code>std_d_ns_time</code>	The standard deviation of the DNS lookup times, calculated from the raw data points for the rollup period.
<code>avg_d_state</code>	The average availability of the web page, calculated from the raw data points for the rollup period. Availability values are either zero (0, unavailable) or one (1, available); average values will range from zero to one.
<code>max_d_state</code>	The value of the single highest availability poll for the web page during the rollup period. Values are either zero (0, unavailable) or one (1, available).

Field	Description
min_d_state	The value of the single lowest availability poll for the web page during the rollup period. Values are either zero (0, unavailable) or one (1, available).
sum_d_state	The sum of all availability values for the web page during the rollup period. Availability values are either zero (0, unavailable) or one (1, available).
std_d_state	The standard deviation of availability values for the web page, calculated from the raw data points for the rollup period.
min_d_trans_time	The lowest transaction time, in seconds, of all polls during the rollup period.
max_d_trans_time	The highest transaction time, in seconds, of all polls during the rollup period.
avg_d_trans_time	The average transaction time, in seconds, calculated from the raw data points for the rollup period.
sum_d_trans_time	The sum of all transaction times, in seconds, during the rollup period.
std_d_trans_time	The standard deviation of the transaction times, calculated from the raw data points for the rollup period.

For the resource `/data_performance_raw/device/monitor_cv`, the `data_fields` option can include the following fields:

Field	Description
d_conn_time	The connection time, in seconds.
d_dl_size	The download size, in bytes.
d_dl_speed	The download speed, in bytes/second.
d_ns_time	The DNS lookup time, in seconds.
d_state	The availability of the web page. Availability values are either zero (0, unavailable) or one (1, available).
d_trans_time	The transaction time, in seconds.

Fields for SOAP/XML Transaction Monitor Resources

For the resource `/data_performance/device/monitor_tv`, the `data_fields` option can include the following fields:

Field	Description
<code>min_d_conn_time</code>	The lowest connection time, in seconds, of all polls during the rollup period.
<code>max_d_conn_time</code>	The highest connection time, in seconds, of all polls during the rollup period.
<code>avg_d_conn_time</code>	The average connection time, in seconds, calculated from the raw data points for the rollup period.
<code>sum_d_conn_time</code>	The sum of all connection times, in seconds, during the rollup period.
<code>std_d_conn_time</code>	The standard deviation of the connection times, calculated from the raw data points for the rollup period.
<code>min_d_dl_size</code>	The lowest download size, in bytes, of all polls during the rollup period.
<code>max_d_dl_size</code>	The highest download size, in bytes, of all polls during the rollup period.
<code>avg_d_dl_size</code>	The average download size, in bytes, calculated from the raw data points for the rollup period.
<code>sum_d_dl_size</code>	The sum of all download sizes, in bytes, during the rollup period.
<code>std_d_dl_size</code>	The standard deviation of the download sizes, calculated from the raw data points for the rollup period.
<code>min_d_dl_speed</code>	The lowest download speed, in bytes/second, of all polls during the rollup period.
<code>max_d_dl_speed</code>	The highest download speed, in bytes/second, of all polls during the rollup period.
<code>avg_d_dl_speed</code>	The average download speed, in bytes/second, calculated from the raw data points for the rollup period.
<code>sum_d_dl_speed</code>	The sum of all download speeds, in bytes/second, during the rollup period.
<code>std_d_dl_speed</code>	The standard deviation of the download speeds, calculated from the raw data points for the rollup period.
<code>min_d_ns_time</code>	The lowest DNS lookup time, in seconds, of all polls during the rollup period.
<code>max_d_ns_time</code>	The highest DNS lookup time, in seconds, of all polls during the rollup period.
<code>avg_d_ns_time</code>	The average DNS lookup time, in seconds, calculated from the raw data points for the rollup period.
<code>sum_d_ns_time</code>	The sum of all DNS lookup times, in seconds, during the rollup period.
<code>std_d_ns_time</code>	The standard deviation of the DNS lookup times, calculated from the raw data points for the rollup period.
<code>avg_d_state</code>	The average availability of the web service, calculated from the raw data points for the rollup period. Availability values are either zero (0, unavailable) or one (1, available); average values will range from zero to one.
<code>max_d_state</code>	The value of the single highest availability poll for the web service during the rollup period. Values are either zero (0, unavailable) or one (1, available).

Field	Description
min_d_state	The value of the single lowest availability poll for the web service during the rollup period. Values are either zero (0, unavailable) or one (1, available).
sum_d_state	The sum of all availability values for the web service during the rollup period. Availability values are either zero (0, unavailable) or one (1, available).
std_d_state	The standard deviation of availability values for the web service, calculated from the raw data points for the rollup period.
min_d_trans_time	The lowest transaction time, in seconds, of all polls during the rollup period.
max_d_trans_time	The highest transaction time, in seconds, of all polls during the rollup period.
avg_d_trans_time	The average transaction time, in seconds, calculated from the raw data points for the rollup period.
sum_d_trans_time	The sum of all transaction times, in seconds, during the rollup period.
std_d_trans_time	The standard deviation of the transaction times, calculated from the raw data points for the rollup period.

For the resource `/data_performance_raw/device/monitor_tv`, the `data_fields` option can include the following fields:

Field	Description
d_conn_time	The connection time, in seconds.
d_dl_size	The download size, in bytes.
d_dl_speed	The download speed, in bytes/second.
d_ns_time	The DNS lookup time, in seconds.
d_state	The availability of the web service. Availability values are either zero (0, unavailable) or one (1, available).
d_trans_time	The transaction time, in seconds.

Fields for Process Monitor Resources

For the resource `/data_performance/device/monitor_process`, the `data_fields` option can include the following fields:

Field	Description
min_d_check	The average availability of the process, calculated from the raw data points for the rollup period. Availability values are either zero (0, valid process is not running or illicit process is running) or one (1, valid process is running or illicit process is not running); average values will range from zero to one.
max_d_check	The value of the single highest availability poll for the process during the rollup period. Values are either zero (0, valid process is not running or illicit process is running) or one (1, valid process is running or illicit process is not running).
avg_d_check	The value of the single lowest availability poll for the process during the rollup period. Values are either zero (0, valid process is not running or illicit process is running) or one (1, valid process is running or illicit process is not running).
sum_d_check	The sum of all availability values for the process during the rollup period.

Field	Description
	Availability values are either zero (0, valid process is not running or illicit process is running) or one (1, valid process is running or illicit process is not running).
std_d_check	The standard deviation of availability values for the process, calculated from the raw data points for the rollup period.
min_d_counter	The average number of instances of the process, calculated from the raw data points for the rollup period.
max_d_counter	The number of instances of the process at the single poll with the highest value.
avg_d_counter	The number of instances of the process at the single poll with the lowest value.
sum_d_counter	The sum of the number of instances of the process running at each poll during the rollup period.
std_d_counter	The standard deviation of number of instances of the process running, calculated from the raw data points for the rollup period.

For the resource `/data_performance_raw/device/monitor_process`, the `data_fields` option can include the following fields:

Field	Description
d_check	The availability of the process. Availability values are either zero (0, valid process is not running or illicit process is running) or one (1, valid process is running or illicit process is not running).
d_counter	The number of instances of the processes running.

Fields for Windows Service Monitor Resources

For the resource `/data_performance/device/monitor_service`, the `data_fields` option can include the following fields:

Field	Description
avg_d_state	The average availability of the service, calculated from the raw data points for the rollup period. Availability values are either zero (0, valid service is not running or illicit service is running) or one (1, valid service is running or illicit service is not running); average values will range from zero to one.
max_d_state	The value of the single highest availability poll for the service during the rollup period. Values are either zero (0, valid service is not running or illicit service is running) or one (1, valid service is running or illicit process is not running).
min_d_state	The value of the single lowest availability poll for the service during the rollup period. Values are either zero (0, valid service is not running or illicit service is running) or one (1, valid service is running or illicit service is not running).
sum_d_state	The sum of all availability values for the service during the rollup period. Availability values are either zero (0, valid service is not running or illicit service is running) or one (1, valid service is running or illicit service is not running).
std_d_state	The standard deviation of availability values for the service, calculated from

Field	Description
	the raw data points for the rollup period.

For the resource */data_performance_raw/device/monitor_service*, the `data_fields` option can include the following fields:

Field	Description
<code>d_state</code>	The availability of the service. Availability values are either zero (0, valid service is not running or illicit service is running) or one (1, valid service is running or illicit service is not running).

Fields for Email Round-Trip Monitor Resources

For the resource */data_performance/device/monitor_email*, the `data_fields` option can include the following fields:

Field	Description
<code>min_d_rt_time</code>	The lowest email round-trip time, in seconds, of all polls during the rollup period.
<code>max_d_rt_time</code>	The highest email round-trip time, in seconds, of all polls during the rollup period.
<code>avg_d_rt_time</code>	The average email round-trip time, in seconds, calculated from the raw data points for the rollup period.
<code>sum_d_rt_time</code>	The sum of all email round-trip times, in seconds, during the rollup period.
<code>std_d_rt_time</code>	The standard deviation of the email round-trip times, calculated from the raw data points for the rollup period.
<code>min_d_state</code>	The value of the single lowest availability poll for the mail process during the rollup period. Values are either zero (0, email response was not received within the threshold time) or one (1, email response was received within the threshold time).
<code>max_d_state</code>	The value of the single highest availability poll for the mail process during the rollup period. Values are either zero (0, email response was not received within the threshold time) or one (1, email response was received within the threshold time).
<code>avg_d_state</code>	The average availability of the mail process, calculated from the raw data points for the rollup period. Availability values are either zero (0, email response was not received within the threshold time) or one (1, email response was received within the threshold time); average values will range from zero to one.
<code>sum_d_state</code>	The sum of all availability values for the mail process during the rollup period. Availability values are either zero (0, email response was not received within the threshold time) or one (1, email response was received within the threshold time).
<code>std_d_state</code>	The standard deviation of availability values for the mail process, calculated from the raw data points for the rollup period.

For the resource */data_performance_raw/device/monitor_email*, the `data_fields` option can include the following fields:

Field	Description
<code>d_rt_time</code>	The email round-trip time, in seconds.
<code>d_state</code>	The availability of the mail service. Availability values are either zero (0, email response was not received within the threshold time) or one (1, email response was received within the threshold time).

Fields for DNS Monitor Resources

For the resource */data_performance/device/monitor_dns*, the `data_fields` option can include the following fields:

Field	Description
<code>min_d_ns_time</code>	The lowest DNS lookup time, in seconds, of all polls during the rollup period.
<code>max_d_ns_time</code>	The highest DNS lookup time, in seconds, of all polls during the rollup period.
<code>avg_d_ns_time</code>	The average DNS lookup time, in seconds, calculated from the raw data points for the rollup period.
<code>sum_d_ns_time</code>	The sum of all DNS lookup times, in seconds, during the rollup period.
<code>std_d_ns_time</code>	The standard deviation of the DNS lookup times, calculated from the raw data points for the rollup period.
<code>min_d_state</code>	The value of the single lowest availability poll for the DNS record during the rollup period. Values are either zero (0, no response or DNS record does not match the policy) or one (1, DNS record matches the policy).
<code>max_d_state</code>	The value of the single highest availability poll for the DNS record during the rollup period. Values are either zero (0, no response or DNS record does not match the policy) or one (1, DNS record matches the policy).
<code>avg_d_state</code>	The average availability of the DNS record, calculated from the raw data points for the rollup period. Availability values are either zero (0, no response or DNS record does not match the policy) or one (1, DNS record matches the policy); average values will range from zero to one.
<code>sum_d_state</code>	The sum of all availability values for the DNS record during the rollup period. Availability values are either zero (0, no response or DNS record does not match the policy) or one (1, DNS record matches the policy).
<code>std_d_state</code>	The standard deviation of availability values for the DNS record, calculated from the raw data points for the rollup period.

For the resource */data_performance_raw/device/monitor_dns*, the `data_fields` option can include the following fields:

Field	Description
<code>d_ns_time</code>	The DNS lookup time, in seconds.
<code>d_state</code>	The availability of the DNS record. Availability values are either zero (0, no response or DNS record does not match the policy) or one (1, DNS record matches the policy).

Fields for File System Resources

For the resource */data_performance/device/filesystem*, the `data_fields` option can include the following fields:

Field	Description
<code>min_d_used</code>	The lowest file system usage, in kilobytes, of all polls during the rollup period.
<code>max_d_used</code>	The highest file system usage, in kilobytes, of all polls during the rollup period.
<code>avg_d_used</code>	The average file system usage, in kilobytes, calculated from the raw data points for the rollup period.
<code>sum_d_used</code>	The sum of file system usage values, in kilobytes, during the rollup period.
<code>std_d_used</code>	The standard deviation of the file system usage values, calculated from the raw data points for the rollup period.
<code>min_d_used_percent</code>	The lowest file system utilization, in percent, of all polls during the rollup period.
<code>max_d_used_percent</code>	The highest file system utilization, in percent, of all polls during the rollup period.
<code>avg_d_used_percent</code>	The average file system utilization, in percent, calculated from the raw data points for the rollup period.
<code>sum_d_used_percent</code>	The sum of all file system utilization values, in percent, during the rollup period.
<code>sum_d_used_percent</code>	The standard deviation of the file system usage values, calculated from the raw data points for the rollup period.

For the resource */data_performance_raw/device/filesystem*, the `data_fields` option can include the following fields:

Field	Description
<code>d_used</code>	File system usage in kilobytes.
<code>d_used_percent</code>	File system utilization in percent.

Fields for Availability Resources

For the resource */data_performance/device/avail*, the `data_fields` option can include the following fields:

Field	Description
<code>min_d_check</code>	The value of the single lowest availability poll for the device during the rollup period. Values are either zero (0, unavailable) or one (1, available).
<code>max_d_check</code>	The value of the single highest availability poll for the device during the rollup period. Values are either zero (0, unavailable) or one (1, available).
<code>avg_d_check</code>	The average availability of the device, calculated from the raw data points for the rollup period. Availability values are either zero (0, unavailable) or one (1, available); average values will range from zero to one.
<code>sum_d_check</code>	The sum of all availability values for the device during the rollup period. Availability values are either zero (0, unavailable) or one (1, available).
<code>std_d_check</code>	The standard deviation of availability values for the device, calculated from the raw data points for the rollup period.
<code>min_d_latency</code>	The value of the single lowest latency poll, in milliseconds, for the device during the rollup period.
<code>max_d_latency</code>	The value of the single highest latency poll, in milliseconds, for the device during the rollup period.
<code>avg_d_latency</code>	The average latency of the device, in milliseconds, calculated from the raw data points for the rollup period.
<code>sum_d_latency</code>	The sum of all latency values, in milliseconds, for the device during the rollup period.
<code>std_d_latency</code>	The standard deviation of latency values for the device, calculated from the raw data points for the rollup period.

For the resource */data_performance_raw/device/avail*, the `data_fields` option can include the following fields:

Field	Description
<code>d_check</code>	The availability of the device. Availability values are either zero (0, unavailable) or one (1, available).
<code>d_latency</code>	The latency of the device, in milliseconds.

Fields for Interface Resources

For the resource `/data_performance/interface`, the `data_fields` option can include the following fields for utilization, error, and discard metrics:

NOTE: A single request to `/data_performance/interface` cannot include data fields from this list and data fields for packet metrics.

Field	Description
<code>min_d_bps_in</code>	The lowest inbound bandwidth, in bits per second, per poll for the interface during the rollup period.
<code>max_d_bps_in</code>	The highest inbound bandwidth, in bits per second, per poll for the interface during the rollup period.
<code>avg_d_bps_in</code>	The average inbound bandwidth, in bits per second, per poll for the interface during the rollup period.
<code>sum_d_bps_in</code>	The total number of discarded inbound bandwidth, in bits per second, per poll for the interface during the rollup period.
<code>std_d_bps_in</code>	The standard deviation of discarded inbound bandwidth, in bits per second, per poll for the interface during the rollup period.
<code>min_d_bps_out</code>	The lowest number of discarded outbound bandwidth, in bits per second, per poll for the interface during the rollup period.
<code>max_d_bps_out</code>	The highest number of discarded outbound bandwidth, in bits per second, per poll for the interface during the rollup period.
<code>avg_d_bps_out</code>	The average number of discarded outbound bandwidth, in bits per second, per poll for the interface during the rollup period.
<code>sum_d_bps_out</code>	The total number of discarded outbound bandwidth, in bits per second, per poll for the interface during the rollup period.
<code>std_d_bps_out</code>	The standard deviation of discarded outbound bandwidth, in bits per second, per poll for the interface during the rollup period.
<code>min_d_discards_in</code>	The lowest number of discarded inbound packets per poll for the interface during the rollup period.
<code>min_d_discards_in</code>	The lowest number of discarded inbound packets per poll for the interface during the rollup period.
<code>max_d_discards_in</code>	The highest number of discarded inbound packets per poll for the interface during the rollup period.
<code>avg_d_discards_in</code>	The average number of discarded inbound packets per poll for the interface during the rollup period.
<code>sum_d_discards_in</code>	The total number of discarded inbound packets for the interface during the rollup period.
<code>std_d_discards_in</code>	The standard deviation of discarded inbound packets for the interface, calculated from the raw data points for the rollup period.

Field	Description
min_d_discards_out	The lowest number of discarded outbound packets per poll for the interface during the rollup period.
max_d_discards_out	The highest number of discarded outbound packets per poll for the interface during the rollup period.
avg_d_discards_out	The average number of discarded outbound packets per poll for the interface during the rollup period.
sum_d_discards_out	The total number of discarded outbound packets for the interface during the rollup period.
std_d_discards_out	The standard deviation of discarded outbound packets for the interface, calculated from the raw data points for the rollup period.
min_d_errors_in	The lowest number of inbound packet errors per poll for the interface during the rollup period.
max_d_errors_in	The highest number of inbound packet errors per poll for the interface during the rollup period.
avg_d_errors_in	The average number of inbound packet errors per poll for the interface during the rollup period.
sum_d_errors_in	The total number of inbound packet errors for the interface during the rollup period.
std_d_errors_in	The standard deviation of inbound packet errors for the interface, calculated from the raw data points for the rollup period.
min_d_errors_out	The lowest number of outbound packet errors per poll for the interface during the rollup period.
max_d_errors_out	The highest number of outbound packet errors per poll for the interface during the rollup period.
avg_d_errors_out	The average number of outbound packet errors per poll for the interface during the rollup period.
sum_d_errors_out	The total number of outbound packet errors for the interface during the rollup period.
std_d_errors_out	The standard deviation of outbound packet errors for the interface, calculated from the raw data points for the rollup period.
min_d_octets_in	The lowest number of inbound octets per poll for the interface during the rollup period.
max_d_octets_in	The highest number of inbound octets per poll for the interface during the rollup period.
avg_d_octets_in	The average number of inbound octets per poll for the interface during the rollup period.
sum_d_octets_in	The total number of inbound octets for the interface during the rollup period.
std_d_octets_in	The standard deviation of inbound octets for the interface, calculated from the raw data points for the rollup period.
min_d_octets_out	The lowest number of outbound octets per poll for the interface during the rollup period.

Field	Description
max_d_octets_out	The highest number of outbound octets per poll for the interface during the rollup period.
avg_d_octets_out	The average number of outbound octets per poll for the interface during the rollup period.
sum_d_octets_out	The total number of outbound octets for the interface during the rollup period.
std_d_octets_out	The standard deviation of outbound octets for the interface, calculated from the raw data points for the rollup period.
min_d_perc_discards_in	The lowest percentage of discarded inbound packets per poll for the interface during the rollup period.
max_d_perc_discards_in	The highest percentage of discarded inbound packets per poll for the interface during the rollup period.
avg_d_perc_discards_in	The average percentage of discarded inbound packets per poll for the interface during the rollup period.
sum_d_perc_discards_in	The sum of all percentages of discarded inbound packets for the interface during the rollup period.
std_d_perc_discards_in	The standard deviation of discarded inbound packets in percent for the interface, calculated from the raw data points for the rollup period.
min_d_perc_discards_out	The lowest percentage of discarded outbound packets per poll for the interface during the rollup period.
max_d_perc_discards_out	The highest percentage of discarded outbound packets per poll for the interface during the rollup period.
avg_d_perc_discards_out	The average percentage of discarded outbound packets per poll for the interface during the rollup period.
sum_d_perc_discards_out	The sum of all percentages of discarded outbound packets for the interface during the rollup period.
std_d_perc_discards_out	The standard deviation of discarded outbound packets in percent for the interface, calculated from the raw data points for the rollup period.
min_d_perc_errors_in	The lowest percentage of inbound packet errors per poll for the interface during the rollup period.
max_d_perc_errors_in	The highest percentage of inbound packet errors per poll for the interface during the rollup period.
avg_d_perc_errors_in	The average percentage of inbound packet errors per poll for the interface during the rollup period.
sum_d_perc_errors_in	The sum of all percentages of inbound packet errors for the interface during the rollup period.
std_d_perc_errors_in	The standard deviation of inbound packet errors in percent for the interface, calculated from the raw data points for the rollup period.
min_d_perc_errors_out	The lowest percentage of outbound packet errors per poll for the interface during the rollup period.
max_d_perc_errors_out	The highest percentage of outbound packet errors per poll for the interface during the rollup period.

Field	Description
avg_d_perc_errors_out	The average percentage of outbound packet errors per poll for the interface during the rollup period.
sum_d_perc_errors_out	The sum of all percentages of outbound packet errors for the interface during the rollup period.
std_d_perc_errors_out	The standard deviation of outbound packet errors in percent for the interface, calculated from the raw data points for the rollup period.
min_d_perc_in	The lowest inbound utilization, in percent, per poll for the interface during the rollup period.
max_d_perc_in	The highest inbound utilization, in percent, per poll for the interface during the rollup period.
avg_d_perc_in	The average inbound utilization, in percent, for the interface during the rollup period.
sum_d_perc_in	The sum of all percentage values for inbound utilization for the interface during the rollup period.
std_d_perc_in	The standard deviation of inbound utilization values for the interface, calculated from the raw data points for the rollup period.
min_d_perc_out	The lowest outbound utilization, in percent, per poll for the interface during the rollup period.
max_d_perc_out	The highest outbound utilization, in percent, per poll for the interface during the rollup period.
avg_d_perc_out	The average outbound utilization, in percent, for the interface during the rollup period.
sum_d_perc_out	The sum of all percentage values for outbound utilization for the interface during the rollup period.
std_d_perc_out	The standard deviation of outbound utilization values for the interface, calculated from the raw data points for the rollup period.

For the resource ***/data_performance/interface***, the `data_fields` option can include the following fields for packet metrics:

NOTE: A single request to ***/data_performance_raw/interface*** cannot include data fields from this list and data fields for utilization, error, and discard metrics.

Field	Description
min_d_ifp_unicast_perc_in	The lowest percentage of inbound unicast packets per poll for the interface during the rollup period.
max_d_ifp_unicast_perc_in	The highest percentage of inbound unicast packets per poll for the interface during the rollup period.
avg_d_ifp_unicast_perc_in	The average percentage of inbound unicast packets per poll for the interface during the rollup period.

Field	Description
sum_d_ifp_unicast_perc_in	The sum of all percentages of inbound unicast packets for the interface during the rollup period.
std_d_ifp_unicast_perc_in	The standard deviation of inbound unicast packets in percent for the interface, calculated from the raw data points for the rollup period.
min_d_ifp_multicast_perc_in	The lowest percentage of inbound multicast packets per poll for the interface during the rollup period.
max_d_ifp_multicast_perc_in	The highest percentage of inbound multicast packets per poll for the interface during the rollup period.
avg_d_ifp_multicast_perc_in	The average percentage of inbound multicast packets per poll for the interface during the rollup period.
sum_d_ifp_multicast_perc_in	The sum of all percentages of inbound multicast packets for the interface during the rollup period.
std_d_ifp_multicast_perc_in	The standard deviation of inbound multicast packets in percent for the interface, calculated from the raw data points for the rollup period.
min_d_ifp_broadcast_perc_in	The lowest percentage of inbound broadcast packets per poll for the interface during the rollup period.
max_d_ifp_broadcast_perc_in	The highest percentage of inbound broadcast packets per poll for the interface during the rollup period.
avg_d_ifp_broadcast_perc_in	The average percentage of inbound broadcast packets per poll for the interface during the rollup period.
sum_d_ifp_broadcast_perc_in	The sum of all percentages of inbound broadcast packets for the interface during the rollup period.
std_d_ifp_broadcast_perc_in	The standard deviation of inbound broadcast packets in percent for the interface, calculated from the raw data points for the rollup period.
min_d_ifp_unicast_perc_out	The lowest percentage of outbound unicast packets per poll for the interface during the rollup period.
max_d_ifp_unicast_perc_out	The highest percentage of outbound unicast packets per poll for the interface during the rollup period.
avg_d_ifp_unicast_perc_out	The average percentage of outbound unicast packets per poll for the interface during the rollup period.
sum_d_ifp_unicast_perc_out	The sum of all percentages of outbound unicast packets for the interface during the rollup period.
std_d_ifp_unicast_perc_out	The standard deviation of outbound unicast packets in percent for the interface, calculated from the raw data points for the rollup period.
min_d_ifp_multicast_perc_out	The lowest percentage of outbound multicast packets per poll for the interface during the rollup period.
max_d_ifp_multicast_perc_out	The highest percentage of outbound multicast packets per poll for the interface during the rollup period.
avg_d_ifp_multicast_perc_out	The average percentage of outbound multicast packets per poll for the interface during the rollup period.
sum_d_ifp_multicast_perc_out	The sum of all percentages of outbound multicast packets for the interface during the rollup period.

Field	Description
perc_out	during the rollup period.
std_d_ifp_multicast_perc_out	The standard deviation of outbound multicast packets in percent for the interface, calculated from the raw data points for the rollup period.
min_d_ifp_broadcast_perc_out	The lowest percentage of outbound broadcast packets per poll for the interface during the rollup period.
max_d_ifp_broadcast_perc_out	The highest percentage of outbound broadcast packets per poll for the interface during the rollup period.
avg_d_ifp_broadcast_perc_out	The average percentage of outbound broadcast packets per poll for the interface during the rollup period.
sum_d_ifp_broadcast_perc_out	The sum of all percentages of outbound broadcast packets for the interface during the rollup period.
std_d_ifp_broadcast_perc_out	The standard deviation of outbound broadcast packets in percent for the interface, calculated from the raw data points for the rollup period.
min_d_ifp_unicast_in	The lowest inbound unicast packet rate (packets/second) per poll for the interface during the rollup period.
max_d_ifp_unicast_in	The highest inbound unicast packet rate (packets/second) per poll for the interface during the rollup period.
avg_d_ifp_unicast_in	The average inbound unicast packet rate (packets/second) per poll for the interface during the rollup period.
sum_d_ifp_unicast_in	The total of all inbound unicast packet rates (packets/second) for the interface during the rollup period.
std_d_ifp_unicast_in	The standard deviation of inbound unicast packet rates (packets/second) for the interface, calculated from the raw data points for the rollup period.
min_d_ifp_multicast_in	The lowest inbound multicast packet rate (packets/second) per poll for the interface during the rollup period.
max_d_ifp_multicast_in	The highest inbound multicast packet rate (packets/second) per poll for the interface during the rollup period.
avg_d_ifp_multicast_in	The average inbound multicast packet rate (packets/second) per poll for the interface during the rollup period.
sum_d_ifp_multicast_in	The total all inbound multicast packet rates (packets/second) for the interface during the rollup period.
std_d_ifp_multicast_in	The standard deviation of inbound multicast packet rates (packets/second) for the interface, calculated from the raw data points for the rollup period.
min_d_ifp_broadcast_in	The lowest inbound broadcast packet rate (packets/second) per poll for the interface during the rollup period.
max_d_ifp_broadcast_in	The highest inbound broadcast packet rate (packets/second) per poll for the interface during the rollup period.
avg_d_ifp_broadcast_in	The average inbound broadcast packet rate (packets/second) per poll for the interface during the rollup period.
sum_d_ifp_broadcast_in	The total of all inbound broadcast rates (packets/second) for the interface during the rollup period.

Field	Description
std_d_ifp_broadcast_in	The standard deviation of inbound broadcast packet rates (packets/second) for the interface, calculated from the raw data points for the rollup period.
min_d_ifp_unicast_out	The lowest outbound unicast packet rate (packets/second) per poll for the interface during the rollup period.
max_d_ifp_unicast_out	The highest outbound unicast packet rate (packets/second) per poll for the interface during the rollup period.
avg_d_ifp_unicast_out	The average outbound unicast packet rate (packets/second) per poll for the interface during the rollup period.
sum_d_ifp_unicast_out	The total of all outbound unicast packet rates (packets/second) for the interface during the rollup period.
std_d_ifp_unicast_out	The standard deviation of outbound unicast packet rates (packets/second) for the interface, calculated from the raw data points for the rollup period.
min_d_ifp_multicast_out	The lowest outbound multicast packet rate (packets/second) per poll for the interface during the rollup period.
max_d_ifp_multicast_out	The highest outbound multicast packet rate (packets/second) per poll for the interface during the rollup period.
avg_d_ifp_multicast_out	The average outbound multicast packet rate (packets/second) per poll for the interface during the rollup period.
sum_d_ifp_multicast_out	The total all outbound multicast packet rates (packets/second) for the interface during the rollup period.
std_d_ifp_multicast_out	The standard deviation of outbound multicast packet rates (packets/second) for the interface, calculated from the raw data points for the rollup period.
min_d_ifp_broadcast_out	The lowest outbound broadcast packet rate (packets/second) per poll for the interface during the rollup period.
max_d_ifp_broadcast_out	The highest outbound broadcast packet rate (packets/second) per poll for the interface during the rollup period.
avg_d_ifp_broadcast_out	The average outbound broadcast packet rate (packets/second) per poll for the interface during the rollup period.
sum_d_ifp_broadcast_out	The total of all outbound broadcast rates (packets/second) for the interface during the rollup period.
std_d_ifp_broadcast_out	The standard deviation of outbound broadcast packet rates (packets/second) for the interface, calculated from the raw data points for the rollup period.

For the resource */data_performance_raw/interface*, the `data_fields` option can include the following fields for utilization, error, and discard metrics:

NOTE: A single request to */data_performance_raw/interface* cannot include data fields from this list and data fields for packet metrics.

Field	Description
d_discards_in	The number of inbound packet discards for an interface.
d_discards_out	The number of outbound packet discards for an interface.
d_errors_in	The number of inbound packet errors for an interface.
d_errors_out	The number of outbound packet errors for an interface.
d_octets_in	The number of inbound octets for an interface.
d_octets_out	The number of outbound octets for an interface.
d_bps_in	The inbound utilization for an interface, in bytes per second.
d_bps_out	The outbound utilization for an interface, in bytes per second.
d_perc_discards_in	The percentage of inbound packets that were discarded for an interface.
d_perc_discards_out	The percentage of outbound packets that were discarded for an interface.
d_perc_errors_in	The percentage of inbound packets that caused errors for an interface.
d_perc_errors_out	The percentage of outbound packets that caused errors for an interface.
d_perc_in	The inbound utilization for an interface, in percent.
d_perc_out	The outbound utilization for an interface, in percent.

For the resource `/data_performance_raw/interface`, the `data_fields` option can include the following fields for packet metrics:

NOTE: A single request to `/data_performance_raw/interface` cannot include data fields from this list and data fields for utilization, error, and discard metrics.

Field	Description
d_ifp_unicast_perc_in	The percentage of inbound packets that were unicast for an interface.
d_ifp_multicast_perc_in	The percentage of inbound packets that were multicast for an interface.
d_ifp_broadcast_perc_in	The percentage of inbound packets that were broadcast for an interface.
d_ifp_unicast_perc_out	The percentage of outbound packets that were unicast for an interface.
d_ifp_multicast_perc_out	The percentage of outbound packets that were multicast for an interface.
d_ifp_broadcast_perc_out	The percentage of outbound packets that were broadcast for an interface.
d_ifp_unicast_in	The number of inbound unicast packets per second for an interface.
d_ifp_multicast_in	The number of inbound multicast packets per second for an interface.
d_ifp_broadcast_in	The number of inbound broadcast packets per second for an interface.
d_ifp_unicast_out	The number of outbound unicast packets per second for an interface.
d_ifp_multicast_out	The number of outbound multicast packets per second for an interface.
d_ifp_broadcast_out	The number of outbound broadcast packets per second for an interface.

Fields for CBQoS Resources

For the resource `/data_performance/cbqos`, the `data_fields` option can include the following fields:

Field	Description
<code>min_classmap_pre_policy_rate_bits_per_second</code>	The lowest interface utilization, in bps, before applying the CBQoS policy during the rollup period.
<code>max_classmap_pre_policy_rate_bits_per_second</code>	The highest interface utilization, in bps, before applying the CBQoS policy during the rollup period.
<code>avg_classmap_pre_policy_rate_bits_per_second</code>	The average interface utilization, in bps, before applying the CBQoS policy during the rollup period.
<code>sum_classmap_pre_policy_rate_bits_per_second</code>	The total interface utilization, in bps, before applying the CBQoS policy during the rollup period.
<code>std_classmap_pre_policy_rate_bits_per_second</code>	The standard deviation of the interface utilization values (calculated from the raw data points for the rollup period), before applying the CBQoS policy .
<code>min_classmap_post_policy_rate_bits_per_second</code>	The lowest interface utilization, in bps, after applying the CBQoS policy during the rollup period.
<code>max_classmap_post_policy_rate_bits_per_second</code>	The highest interface utilization, in bps, after applying the CBQoS policy during the rollup period.
<code>avg_classmap_post_policy_rate_bits_per_second</code>	The average interface utilization, in bps, after applying the CBQoS policy during the rollup period.
<code>sum_classmap_post_policy_rate_bits_per_second</code>	The total interface utilization, in bps, after applying the CBQoS policy during the rollup period.
<code>std_classmap_post_policy_rate_bits_per_second</code>	The standard deviation of the interface utilization, in bps (calculated from the raw data points for the rollup period), after applying the CBQoS policy.
<code>min_classmap_drop_rate_bits_per_second</code>	The lowest drop rate, in bps, for the class map during the rollup period.
<code>max_classmap_drop_rate_bits_per_second</code>	The highest drop rate, in bps, for the class map during the rollup period.
<code>avg_classmap_drop_rate_bits_per_second</code>	The average drop rate, in bps, for the class map during the rollup period.
<code>sum_classmap_drop_rate_bits_per_second</code>	The total drop rate, in bps, for the class map during the rollup period.
<code>std_classmap_drop_</code>	The standard deviation of drop rate values (calculated from the raw data points

Field	Description
rate_bits_per_second	for the rollup period), in bps, for the class map.
min_classmap_pre_policy_inbound_bytes	The lowest inbound interface utilization, in bytes, before applying the CBQoS policy during the rollup period.
max_classmap_pre_policy_inbound_bytes	The highest inbound interface utilization, in bytes, before applying the CBQoS policy during the rollup period.
avg_classmap_pre_policy_inbound_bytes	The average inbound interface utilization, in bytes, before applying the CBQoS policy during the rollup period.
sum_classmap_pre_policy_inbound_bytes	The total inbound interface utilization, in bytes, before applying the CBQoS policy during the rollup period.
std_classmap_pre_policy_inbound_bytes	The standard deviation of inbound interface utilization values (calculated from the raw data points for the rollup period), before applying the CBQoS policy.
min_classmap_post_policy_outbound_bytes	The lowest outbound interface utilization, in bytes, after applying the CBQoS policy during the rollup period.
max_classmap_post_policy_outbound_bytes	The highest outbound interface utilization, in bytes, after applying the CBQoS policy during the rollup period.
avg_classmap_post_policy_outbound_bytes	The average outbound interface utilization, in bytes, after applying the CBQoS policy during the rollup period.
sum_classmap_post_policy_outbound_bytes	The total outbound interface utilization, in bytes, after applying the CBQoS policy during the rollup period.
std_classmap_post_policy_outbound_bytes	The standard deviation of outbound interface utilization values (calculated from the raw data points for the rollup period), after applying the CBQoS policy.
min_policing_conforming_rate_bits_per_second	The lowest collected traffic value, in bps, that conformed to the policing policy during the rollup period.
max_policing_conforming_rate_bits_per_second	The highest collected traffic value, in bps, that conformed to the policing policy during the rollup period.
avg_policing_conforming_rate_bits_per_second	The average collected traffic value, in bps, that conformed to the policing policy during the rollup period.
sum_policing_conforming_rate_bits_per_second	The total collected traffic value, in bps, that conformed to the policing policy during the rollup period.
std_policing_conforming_rate_bits_per_second	The standard deviation of the collected traffic values (calculated from the raw data points for the rollup period) that conformed to the policing policy.
min_classmap_drops_bytes	The lowest drop rate, in bytes, for the class map during the rollup period.
max_classmap_drops_bytes	The highest drop rate, in bytes, for the class map during the rollup period.
avg_classmap_drops_bytes	The average drop rate, in bytes, for the class map during the rollup period.

Field	Description
sum_classmap_drops_bytes	The total drop rate, in bytes, for the class map during the rollup period.
std_classmap_drops_bytes	The standard deviation (calculated from the raw data points for the rollup period) of collected drop rate values.
min_policing_non_conforming_rate_bits_per_second	The lowest collected traffic value, in bps, that did not conform to the policing policy during the rollup period.
max_policing_non_conforming_rate_bits_per_second	The highest collected traffic value, in bps, that did not conform to the policing policy during the rollup period.
avg_policing_non_conforming_rate_bits_per_second	The average collected traffic value, in bps, that did not conform to the policing policy during the rollup period.
sum_policing_non_conforming_rate_bits_per_second	The total collected traffic value, in bps, that did not conform to the policing policy during the rollup period.
std_policing_non_conforming_rate_bits_per_second	The standard deviation of the collected traffic values (calculated from the raw data points for the rollup period) that did not conform to the policing policy.
min_policing_violation_rate_bits_per_second	The lowest collected traffic value, in bps, that violated the policing policy during the rollup period.
max_policing_violation_rate_bits_per_second	The highest collected traffic value, in bps, that violated the policing policy during the rollup period.
avg_policing_violation_rate_bits_per_second	The average collected traffic value, in bps, that violated the policing policy during the rollup period.
sum_policing_violation_rate_bits_per_second	The total collected traffic value, in bps, that violated the policing policy during the rollup period.
std_policing_violation_rate_bits_per_second	The standard deviation of the collected traffic values (calculated from the raw data points for the rollup period) that violated the policing policy.
min_policing_conforming_bytes	The lowest collected traffic value, in bytes, that conformed to the policing policy during the rollup period.
max_policing_conforming_bytes	The highest collected traffic value, in bytes, that conformed to the policing policy during the rollup period.
avg_policing_conforming_bytes	The average collected traffic value, in bytes, that conformed to the policing policy during the rollup period.
sum_policing_conforming_bytes	The total collected traffic value, in bytes, that conformed to the policing policy during the rollup period.
std_policing_conforming_bytes	The standard deviation of the collected traffic values (calculated from the raw data points for the rollup period) that conformed to the policing policy.
min_policing_non_conforming_bytes	The lowest collected traffic value, in bytes, that did not conform to the policing policy during the rollup period.
max_policing_non_conforming_bytes	The highest collected traffic value, in bytes, that did not conform to the policing policy during the rollup period.

Field	Description
conforming_bytes	policy during the rollup period.
avg_policing_non_conforming_bytes	The average collected traffic value, in bytes, that did not conform to the policing policy during the rollup period.
sum_policing_non_conforming_bytes	The total collected traffic value, in bytes, that did not conform to the policing policy during the rollup period.
std_policing_non_conforming_bytes	The standard deviation of the collected traffic values (calculated from the raw data points for the rollup period) that did not conform to the policing policy.
min_policing_violations_bytes	The lowest collected traffic value, in bytes, that violated the policing policy during the rollup period.
max_policing_violations_bytes	The highest collected traffic value, in bytes, that violated the policing policy during the rollup period.
avg_policing_violations_bytes	The average collected traffic value, in bytes, that violated the policing policy during the rollup period.
sum_policing_violations_bytes	The total collected traffic value, in bytes, that violated the policing policy during the rollup period.
std_policing_violations_bytes	The standard deviation of the collected traffic values (calculated from the raw data points for the rollup period) that violated the policing policy.
min_queueing_discards_bytes	The lowest discarded traffic, in bytes, for the queueing policy during the rollup period.
max_queueing_discards_bytes	The highest discarded traffic, in bytes, for the queueing policy during the rollup period.
avg_queueing_discards_bytes	The average discarded traffic, in bytes, for the queueing policy during the rollup period.
sum_queueing_discards_bytes	The total discarded traffic, in bytes, for the queueing policy during the rollup period.
std_queueing_discards_bytes	The standard deviation of collected discarded traffic values (calculated from the raw data points for the rollup period) for the queueing policy.
min_queueing_current_queue_depth_bytes	The lowest queue depth, in bytes, for the queueing policy during the rollup period.
max_queueing_current_queue_depth_bytes	The highest queue depth, in bytes, for the queueing policy during the rollup period.
avg_queueing_current_queue_depth_bytes	The average queue depth, in bytes, for the queueing policy during the rollup period.
sum_queueing_current_queue_depth_bytes	The total queue depth, in bytes, for the queueing policy during the rollup period.
std_queueing_current_queue_depth_bytes	The standard deviation of collected queue depth values (calculated from the raw data points for the rollup period) for the queueing policy.
min_classmap_pre_policy_inbound_utilization_percent	The lowest inbound interface utilization, in percent, before applying the CBQoS policy during the rollup period.
max_classmap_pre_policy_inbound_utilization_percent	The highest inbound interface utilization, in percent, before applying the

Field	Description
policy_inbound_utilization_percent	CBQoS policy during the rollup period.
avg_classmap_pre_policy_inbound_utilization_percent	The average inbound interface utilization, in percent, before applying the CBQoS policy during the rollup period.
sum_classmap_pre_policy_inbound_utilization_percent	The total inbound interface utilization, in percent, before applying the CBQoS policy during the rollup period.
std_classmap_pre_policy_inbound_utilization_percent	The standard deviation of the percent inbound interface utilization values (calculated from the raw data points for the rollup period), before applying the CBQoS policy.
min_classmap_post_policy_outbound_utilization_percent	The lowest outbound interface utilization, in percent, after applying the CBQoS policy during the rollup period.
max_classmap_post_policy_outbound_utilization_percent	The highest outbound interface utilization, in percent, after applying the CBQoS policy during the rollup period.
avg_classmap_post_policy_outbound_utilization_percent	The average outbound interface utilization, in percent, after applying the CBQoS policy during the rollup period.
sum_classmap_post_policy_outbound_utilization_percent	The total outbound interface utilization, in percent, after applying the CBQoS policy during the rollup period.
std_classmap_post_policy_outbound_utilization_percent	The standard deviation of the percent outbound interface utilization values (calculated from the raw data points for the rollup period, after applying the CBQoS policy).
min_queueing_discard_rate_bytes_per_second	The lowest discard rate, in bps, for the queueing policy during the rollup period.
max_queueing_discard_rate_bytes_per_second	The highest discard rate, in bps, for the queueing policy during the rollup period.
avg_queueing_discard_rate_bytes_per_second	The average discard rate, in bps, for the queueing policy during the rollup period.
sum_queueing_discard_rate_bytes_per_second	The total discard rate, in bps, for the queueing policy during the rollup period.
std_queueing_discard_rate_bytes_per_second	The standard deviation of collected discard rate values (calculated from the raw data points for the rollup period) for the queueing policy.

For the resource `/data_performance_raw/cbqos`, the `data_fields` option can include the following fields:

Field	Description
classmap_drop_rate_bits_per_second	The drop rate, in bps, for the class map.
classmap_drops_bytes	The drop rate, in bytes, for the class map.
classmap_pre_policy_	The total interface utilization, in bps, before applying the CBQoS policy.

Field	Description
rate_bits_per_second	
classmap_post_policy_rate_bits_per_second	The total interface utilization, in bps, after applying the CBQoS policy.
classmap_pre_policy_inbound_bytes	The inbound interface utilization, in bps, before applying the CBQoS policy.
classmap_post_policy_outbound_bytes	The outbound interface utilization, in bps, after applying the CBQoS policy.
classmap_pre_policy_inbound_utilization_percent	The inbound interface utilization, in percent, before applying the CBQoS policy.
classmap_post_policy_outbound_utilization_percent	The outbound interface utilization, in percent, after applying the CBQoS policy.
policing_conforming_rate_bits_per_second	The total traffic, in bps, that conformed to the policing policy.
policing_non_conforming_rate_bits_per_second	The total traffic, in bps, that did not conform to the policing policy.
policing_violation_rate_bits_per_second	The total traffic, in bps, that violated the policing policy.
policing_conforming_bytes	The total traffic, in bytes, that conformed to the policing policy.
policing_non_conforming_bytes	The total traffic, in bytes, that did not conform to the policing policy.
policing_violations_bytes	The total traffic, in bytes, that violated the policing policy.
queueing_discards_bytes	The discarded traffic, in bytes, for the queueing policy.
queueing_current_queue_depth_bytes	The queue depth, in bytes, for the queueing policy.
queueing_discard_rate_bytes_per_second	The discard rate, in bps, for the queueing policy.

Requesting Data for Specific Devices or Interfaces

By default, the sub-resources of **/data_performance** and **/data_performance_raw** return data for all devices or interfaces for which data of the specified type exists.

The default response from the sub-resources of **/data_performance** and **/data_performance_raw** include a searchspec section. The *fields* section of the searchspec includes a list of attribute values for devices or interfaces. The attribute values can be used to filter the result. See the chapter on [Resources and URIs](#) for information on how to use these fields to filter results.

Filtering Device Resources

The following fields can be used to filter **device** data:

Field	Description
device	Supply numeric values to match against device IDs. The response will include devices with matching IDs.
device/class_type/class	Supply string values to match against Device Class (typically vendors or manufacturers, e.g. "Cisco Systems"). The response will include devices with a matching device class.
device/class_type/description	Supply string values to match against Device Class descriptions (typically device models, e.g. "Catalyst 3750"). The response will include devices with a matching device class.
device/class_type/device_category/cat_name	Supply string values to match against device categories, e.g. "Network.Switches". The response will include devices with a matching device category.
device/class_type/device_category/guid	Supply string values to match against device category GUIDs. The response will include devices with a matching device category.
device/class_type/guid	Supply string values to match against device class GUIDs. The response will include devices with a matching device class.
device/merged_class_type/class	Supply one or more Device Classes (typically vendors or manufacturers, e.g. "Cisco Systems") to filter on. The response will include physical devices that are merged with a component device and that component device has matching device class.
device/merged_class_type/description	Supply string values to match against Device Class descriptions (typically device models, e.g. "Catalyst 3750"). The response will include physical devices that are merged with a component device and that component device has matching device class.
device/merged_class_type/device_category/cat_name	Supply string values to match against device categories, e.g. "Network.Switches". The response will include physical devices that are merged with a component device and that component device has matching device category.
device/merged_class_type/device_category/guid	Supply string values to match against device category GUIDs. The response will include physical devices that are merged with a component device and that component device has matching device category.
device/merged_class_type/guid	Supply string values to match against device class GUIDs. The response will include physical devices that are merged with a component device and that component device has matching device class.
device/name	Supply string values to match against device names. The response will include devices with matching names.
device/organization	Supply URIs to match against organizations. The response will include devices in the matching organizations.
device/organization/company	Supply string values to match against organization names. The response will include devices in the matching organizations.

Field	Description
idx	Supply numeric values to match against: - For Dynamic Applications, the index values assigned to each time series. The response will include time series with matching indexes. - For monitoring policies, the policy IDs. The response will include monitoring policies with matching IDs. - For file systems, the file system IDs. The response will include file systems with matching IDs. This option is not available for the resources <i>/data_performance/device/avail</i> or <i>/data_performance_raw/device/avail</i>.
idx_label	Supply string values to match against: - For Dynamic Applications, the labels associated with each time series. The response will include time series with matching labels. - For Windows service and system process monitoring policies, the name of the service or process. The response will include policies with matching service or process names. - For email round-trip, web content, and SOAP/XML transaction policies, the name of the policy. The response will include policies with matching names. - For DNS monitoring policies, the DNS record. The response will include policies that monitor matching DNS records. - For port monitoring policies, the IP address and port number in the format <i>ip:port</i>. The response will include policies that monitor a matching <i>ip:port</i> string. - For file systems, the file system names. The response will include file systems with matching names. This option is not available for the resources <i>/data_performance/device/avail</i> or <i>/data_performance_raw/device/avail</i>.
ip	This option applies only to the resources <i>/data_performance/device/monitor_port</i> and <i>/data_performance_raw/device/monitor_port</i>. Supply string values to match against policy IP addresses. The response will include monitoring policies with matching IP addresses.
port	This option applies only to the resources <i>/data_performance/device/monitor_port</i> and <i>/data_performance_raw/device/monitor_port</i>. Supply numeric values to match against policy port numbers. The response will include monitoring policies with matching port numbers.

Filtering Interface Resources

The following fields can be used to filter ***interface*** data:

Field	Description
device	Supply numeric values to match against device IDs. The response will include interfaces associated with devices with matching IDs.
device/class_type/class	Supply string values to match against Device Classes (typically vendors or manufacturers, e.g. "Cisco Systems"). The response will include interfaces associated with devices with a matching device class.
device/class_type/description	Supply string values to match against Device Class descriptions (typically device models, e.g. "Catalyst 3750"). The response will include interfaces associated with devices with a matching device class.
device/class_type/device_category/cat_name	Supply string values to match against device categories, e.g. "Network.Switches". The response will include interfaces associated with devices with a matching device category.
device/class_type/device_category/guid	Supply string values to match against device category GUIDs. The response will include interfaces associated with devices with a matching device category.
device/class_type/guid	Supply string values to match against device class GUIDs. The response will include interfaces associated with devices with a matching device class.
device/merged_class_type/class	Supply one or more Device Classes (typically vendors or manufacturers, e.g. "Cisco Systems") to filter on. The response will include interfaces associated with physical devices that are merged with a component device and that component device has matching device class.
device/merged_class_type/description	Supply string values to match against Device Class descriptions (typically device models, e.g. "Catalyst 3750"). The response will include interfaces associated with physical devices that are merged with a component device and that component device has matching device class.
device/merged_class_type/device_category/cat_name	Supply string values to match against device categories, e.g. "Network.Switches". The response will include interfaces associated with physical devices that are merged with a component device and that component device has matching device category.
device/merged_class_type/device_category/guid	Supply string values to match against device category GUIDs. The response will include interfaces associated with physical devices that are merged with a component device and that component device has matching device category.
device/merged_class_type/guid	Supply string values to match against device class GUIDs. The response will include interfaces associated with physical devices that are merged with a component device and that component device has matching device class.
device/name	Supply string values to match against device names. The response will include interfaces associated with devices with matching names.
device/organization	Supply URIs to match against organizations. The response will include interfaces associated with devices in the matching organizations.
device/organization/company	Supply string values to match against organization names. The response will include interfaces associated with devices in the matching organizations.

Field	Description
interface	Supply numeric values to match against interface IDs. The response will include interfaces with matching IDs.
interface/alias	Supply string values to match against interface aliases. The response will include interfaces with matching aliases.
interface/ifDescr	Supply string values to match against interface descriptions. The response will include interfaces with matching descriptions.
interface/name	Supply string values to match against interface names. The response will include interfaces with matching names.
interface/organization	Supply string values to match against organization names. The response will include interfaces in the matching organizations.
interface/tag	Supply string values to match against interface tags. The response will include interfaces with matching tags.

Filtering CBQoS Resources

The [fields listed for Interface resources](#) can also be used to filter CBQoS data. The following CBQoS-specific fields can be used to filter **cbqos** data:

Field	Description
cbqos_object	Supply numeric values to match against CBQoS object IDs. The response will include CBQoS metrics associated with objects with matching IDs. CBQoS objects can be searched and filtered using the <code>/api/cbqos_object</code> resource index.
cbqos_object/name	Supply string values to match against CBQoS object names. The response will include CBQoS metrics associated with objects with matching names.
cbqos_object/type	Supply numeric values to match against CBQoS object types. The response will include CBQoS metrics associated with objects that have a matching type. CBQoS object types can be searched and filtered using the <code>/api/cbqos_type</code> resource index.

Additional Options

The default response from the sub-resources of `/data_performance` and `/data_performance_raw` include a `searchspec` section. The `options` section of the `searchspec` includes a list of options that can be included in a request. The following option appears in addition to the options described in the [Specifying the Time Range for a Data Request](#) section:

- **hide_filterinfo**. If this option is set to 1 in the URI, the response will contain only the `result_set`; the response will not include the `searchspec` section.

Responses from Bulk Performance Data Resources

The response from sub-resources of */data_performance* and */data_performance_raw* that return performance data include one of the following:

- An HTTP 400 response code and a human-readable message indicating required options were not included.
- An HTTP 500 response code and a human-readable message indicating that the appliance servicing the request does not have sufficient memory to generate the requested data set. If you receive this response, you must split your request in to multiple smaller requests.
- Zero or more data sets that match the options specified in the request. A data set is represented as a JSON array or an XML structure bounded by <dataset> tags. Each data set represents:
 - For interface data, the data from a single interface.
 - For CBQoS data, the data from a single interface for a single CBQoS object.
 - For Dynamic Application data, the data for a single index (time series) for a device.
 - For all other device data, the data from a single device.

Each data set includes:

- The URI of the device, interface (where applicable), and CBQoS metric (where applicable) the data set is associated with.
- If applicable, the index and index label for the data series.
- The list of field names included for each data point in the data set. This list of fields will include the time stamp field and the data fields specified in the request options.
- A list of data points. Each data point is a list that includes an entry for each field (time stamp and data fields). To improve performance, the field names are not included with each data point. The field order for each data point matches the list of field names that appears at the beginning of each data set.

Best Practices for Requesting Bulk Performance Data

Overview

This chapter describes the best practices that ScienceLogic recommends you follow when requesting performance data in bulk. This chapter describes how to use these resources to request performance data in bulk.

ScienceLogic offers REST APIs for extraction of collected performance data in bulk: ***data_performance*** and ***data_performance_raw***. These APIs provide a flexible way to extract data from multiple devices and network interfaces for use in downstream systems. The potentially large volume of data and extensive database queries mean that it is very important to follow best practice with these APIs, particularly for ***data_performance_raw***.

1. Work should be spread across multiple concurrent client workers.

Reason: Each ***data_performance_raw*** request that fetches data for multiple devices will be serial internally to Skylar One (i.e. request data for the first matching device, then the second, etc). Splitting the requests and running them concurrently will result in a faster overall response time.

2. Use longer timespans in requests as these are generally more efficient than shorter spans

Reason: Each request to ***data_performance_raw*** includes overhead to check for the existence of raw data tables, apply filters to see what devices and/or series match, etc. This overhead is consistent, regardless of the time-span length of the request. Secondly, the SQL for timeseries data ends up being a range scan using the `collection_time` index - it is a sequential read.

3. If possible, use administrative accounts for requests since they will generally return data faster and with less overhead from the Database Server than user accounts.

Reason: Administrative accounts can leverage an optimization where all multi-tenancy predicates are left out of the queries. In cases where a user has very limited visibility to a small number of elements, a user account is faster, but for bulk data extraction across multiple organizations, an administrative account is faster and more efficient.

4. "Chunking" should be done by element ID directly in the filter by date range and explicit filters. For example:
 - Interfaces: https://API_HOST/api/data_performance_raw/interface?beginstamp=1477537442&data_fields=d_discards_in,d_discards_out,d_errors_in,d_errors_out,d_octets_in,d_octets_out,d_util_in,d_util_out&endstamp=147755442&extended_fetch=1&filter.interface.in=52290,52291,52292,52293,52295&hide_filterinfo=1
 - Dynamic Applications: https://API_HOST/api/data_performance_raw/device/dynamic_app?presentation_objects=B224D22664610BC325EA1929DF3D2325&collection=FA9707E3F4F286D3B267C6DAF3CC4000&duration=24h&filter.0.device.in=1,2,3,4,5,101,202,303&endstamp=now
5. Tune collection according to data set where possible.
 - If the data in question is generally at a lower polling frequency, then increase the size of the window of data being pulled (i.e. 15 hours of 15-minute data = the same number of datapoints per series as 1 hour of 1 minute data).
 - When fetching multiple Dynamic Application presentations at once, group them by application (i.e. presentation_objects=a,b,c where "a,b,c" is a comma-delimited list of presentation GUIDs for the same application).

Example

1

Using the Ticket Resource

Overview

In Skylar One, a ***ticket*** is a request for work. This request can be in response to a problem that needs to be fixed, for routine maintenance, or for any type of work required by your enterprise. A ticket can be assigned to a specific user, to inform and remind that user of requests for work.

This chapter describes how to use the API to perform some basic tasks for managing tickets.

This chapter covers the following topics:

<i>Requirements</i>	100
<i>Getting Started</i>	100
<i>Connecting to the API</i>	101
<i>Viewing a List of Tickets</i>	107
<i>Viewing a List of Tickets and Ticket Details</i>	114
<i>Filtering a List of Tickets</i>	116
<i>Retrieving Information about a Specific Ticket</i>	117
<i>Updating a Ticket</i>	120
<i>Creating a New Ticket</i>	126
<i>Viewing Notes for a Ticket</i>	137
<i>Adding a Note to a Ticket</i>	141
<i>Viewing the Attachments for a Ticket</i>	144

<i>Adding an Attachment to a Ticket Note</i>	<i>150</i>
--	------------

Requirements

- This chapter assumes that you have a working version of cURL installed and can run cURL from a command prompt. For information on cURL, see <http://curl.haxx.se/>.
- To connect to the API, **you must use HTTPS**. If you have not installed or configured a security certificate or if your appliance uses a self-signed certificate, you must use include the "-k" option each time you execute cURL. The "-k" option tells cURL to perform the HTTPS connection without checking the security certificate.
- Through the API, you can perform only actions for which you have permission in Skylar One. To perform the tasks in this chapter, you must connect to the API using an account (username and password), that account must have Access Keys that grant the following:
 - View tickets and ticket details
 - View Ticket Queues
 - Edit a ticket
 - Create a ticket
 - Assign a ticket to a user
 - Add a new note to a ticket

Getting Started

- In the examples in this chapter, we will connect to the example Administration Portal with the IP address of 192.168.10.205. To run these examples on your system, you should replace this IP address with the base URI of the API on the appliance you are using.
- In the examples in this chapter, we will connect to the API using the default account "em7admin" with the example password "examplepassword". To run these examples on your system, you should replace this username and password with a valid username and password for your system.
- In the examples in this chapter, we will execute each HTTP request at a shell prompt or command prompt. However, you can include these requests in a script or program.

CAUTION: The examples in this chapter use the custom-header option "***X-em7-beautify-response:1***". This header tells the API to include white-space in a response, to make it easier to read. However, this header can greatly increase the amount of time required to process a request. ***ScienceLogic recommends you use this header only when testing requests. ScienceLogic strongly discourages you from using this header in integration code.***

Connecting to the API

To connect to the API and view the root directory (with an HTTP GET request), enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api"
```

- **curl -v**. Executes the cURL request. The -v option tells cURL to use verbose mode (displays all header information and all status and error messages). In the response, lines that start with ">" include header data returned by cURL. Lines that start with "<" include header data received by cURL.
- **-H 'X-em7-beautify-response:1'**. The -H option tells cURL to include an additional header in the request. In this case, we're including a ScienceLogic custom header that tells the API to include white-space in the response.
- **-u 'em7admin:examplepassword'**. The -u option tells cURL to authenticate as a specified user. In our example, we authenticated as the user "em7admin" with the password "examplepassword".
- **"https://192.168.10.205/api"**. Connect to the specified URL. In our example, we connected to the API at 192.168.10.205.

The response will look like this (however, we've added line numbers for reference):

```
1) * About to connect() to 192.168.10.205 port 443 (#0)
2) * Trying 192.168.10.205... connected
3) * Connected to 192.168.10.205 (192.168.10.205) port 443 (#0)
4) * Server auth using Basic with user 'em7admin'
5) > GET / HTTP/1.1
6) > Authorization: Basic ZW03YWRTaW46ZW03YWRTaW4=
7) > User-Agent: curl/7.16.3 (powerpc-apple-darwin9.0) libcurl/7.16.3
  OpenSSL/0.9.7l zlib/1.2.3
8) > Host: 192.168.10.205
9) > Accept: */*
10) > X-em7-beautify-response:1
11) >
12) < HTTP/1.1 200 OK
```

```
13) < Date: Wed, 25 Aug 2010 15:47:40 GMT
14) < Server: Apache
15) < X-EM7-Implemented-methods: GET
16) < X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow
17) < Content-Length: 1451
18) < Content-Type: application/json
19) <
20) [
21) {
22) "URI":"\\/account",
23) "description":"Get\\/Update\\/Add\\/Delete User Accounts"
24) },
25) {
26) "URI":"\\/alert",
27) "description":"Create Alerts"
28) },
29) {
30) "URI":"\\/credential",
31) "description":"View Credentials"
32) },
33) {
34) "URI":"\\/device?limit=100",
35) "description":"Get\\/Update\\/Add\\/Delete Devices and Get Collected
Data"
```

```
36) },
37) {
38) "URI":"\\device_group?limit=100",
39) "description":"Get\\Update\\Add\\Delete Device Groups"
40) },
41) {
42) "URI":"\\device_template?limit=100",
43) "description":"Get\\Update\\Add\\Delete Device Templates"
44) },
45) {
46) "URI":"\\discovery_session?limit=100",
47) "description":"Get\\Update\\Add\\Delete Device Discovery Sessions"
48) },
49) {
50) "URI":"\\discovery_session_active?limit=100\\/",
51) "description":"View\\Start\\Stop Active Device Discovery Sessions"
52) },
53) {
54) "URI":"\\dynamic_app\\/",
55) "description":"Get Dynamic Application Resources"
56) },
57) {
58) "URI":"\\event",
59) "description":"View\\Update\\Clear Events"
```

```
60) },
61) {
62) "URI":"\\monitor",
63) "description":"Get\\Update\\Add\\Delete Monitor Policies"
64) },
65) {
66) "URI":"\\organization",
67) "description":"Get\\Update\\Add\\Delete Organizations"
68) },
69) {
70) "URI":"\\ticket?limit=100",
71) "description":"Get\\Update\\Add\\Delete Tickets"
72) },
73) {
74) "URI":"\\ticket_queue",
75) "description":"Get Ticket Queues"
76) }
77) {
78) "URI":"\\ticket_state?limit=100",
79) "description":"Get\\Update\\Add\\Delete Custom Ticket States"
80) },
81) ]
82) Connection #0 to host 192.168.10.205 left intact
83) Closing connection #0
```


- Lines 1-4 show cURL trying to connect to and authenticate with the API.
- Lines 5-11 show the HTTP GET request we sent. The initial request performs a GET on the root directory of the API.
 - ***accept: */****. Specifies that we will use the default accept header. The accept header tells the API how to format the response. The API can respond in XML or JSON. Because we didn't specify an accept header, the API will use the default format, which is JSON. If you want to view the response in XML, you can include the header option "-H 'Accept:application/xml'" in the cURL command.
 - ***X-em7-beautify-response:1***. Tells the API to include white-space in the response, for easier reading.
- Line 12 shows the HTTP version and the HTTP status code for the response.
- Lines 12-19 show the header information for the response.
- Lines 20-81 display the response to the HTTP GET request on the root directory of the API.

The response for the HTTP GET request displays a list of resources. A ***resource*** is a functional area in Skylar One that you can access through the API.

You can interact with the following entities through the API:

- ***Accounts***
- ***Account Lockouts***
- ***Alerts***
- ***Appliances***
- ***Assets***
- ***Collector Groups***
- ***CBQoS Objects***
- ***Collection Labels***
- ***Credentials***
- ***Custom Attributes***
- ***Dashboards***
- ***Devices***
- ***Device Categories***
- ***Device Classes***
- ***Device Interfaces***
- ***Device Groups***
- ***Device Relationships***
- ***Device Templates***
- ***Discovery Sessions***
- ***Dynamic Applications***
- ***Events***

- *Event Categories*
- *External Contacts*
- *File Uploads*
- *Interfaces*
- *Monitoring Policies*
- *Organizations*
- *Performance Data*
- *PowerPacks*
- *Product SKUs*
- *Schedules*
- *System Patches*
- *System Settings*
- *Tasks*
- *System Thresholds*
- *Themes*
- *Thresholds*
- *Tickets*
- *Ticket Categories*
- *Ticket Chargeback*
- *Ticket Logs*
- *Ticket Notes*
- *Ticket Queues*
- *Ticket States*
- *User Policies*
- *Vendors*

For each resource, the response displays the associated URI for accessing the resource and a description that lists the actions you can perform on the resource.

For our example, we'll be looking at the ***ticket*** resource. The ticket resource uses the following URI and includes the following description:

```

69) {
70)  "URI": "\/ticket?limit=100",
71)  "description": "Get\/Update\/Add\/Delete Tickets"
72) }
```

NOTE: The response is in JSON format. Notice that the URI for the ticket includes escaped forward slash characters ("\/").

Viewing a List of Tickets

In the previous section, we used an HTTP GET request to retrieve information about the root directory of the API. The response included a list of resources. From the previous response, we learned that we can retrieve information about tickets.

To access a resource, like *ticket*, we can append its URI to the URI of the root directory. So to access the resource *ticket*, we could enter the following at the command line.

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/ticket"
```

The response looks like this:

```
* About to connect() to 192.168.10.205 port 443 (#0)
* Trying 192.168.10.205... connected
* Connected to 192.168.10.205 (192.168.10.205) port 443 (#0)
* Server auth using Basic with user 'em7admin'
> GET /ticket HTTP/1.1
> Authorization: Basic ZW03YWRTaW46ZW03YWRTaW4=
> User-Agent: curl/7.16.3 (powerpc-apple-darwin9.0) libcurl/7.16.3
OpenSSL/0.9.7l zlib/1.2.3
> Host: 192.168.10.205
> Accept: */*
> em7-beautify-response:1
>
< HTTP/1.1 302 Found
< Date: Wed, 25 Aug 2010 15:48:40 GMT
```

```
< Server: Apache

< X-EM7-Implemented-methods: GET,POST

< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow

< Location: /api/ticket?limit=100

< X-EM7-status-message: ticket index requires a limit

< X-EM7-status-code: FOUND

< Content-Length: 833

< Content-Type: application/json

<

{"searchspec":

{"fields":

{"data":["class","severity","status","source","date_create","date_
update","assigned_
to","resolution","cause","escalation","chargeback","date_close","auto_
close","organization","description","opened_by","updated_by","closed_
by","ticket_queue","parent_ticket"]},

"options":

{

"extended_fetch":

{"type":"boolean","description":"Fetch entire resource if 1 (true), or
resource link only if 0 (false)","default":"0"},

"hide_filterinfo":

{"type":"boolean","description":"Suppress filterspec and current filter
info if 1 (true)","default":"0"},

"limit":
```

```
{
  "type": "int",
  "description": "Number of records to retrieve",
  "default": "100"
},

"offset":

{
  "type": "int",
  "description": "Specifies the index of the first returned resource within the entire result set",
  "default": "0"
}

},

{
  "total_matched": "102",
  "total_returned": 0,
  "result_set": []
}

* Connection #0 to host 192.168.10.205 left intact

* Closing connection #0
```

The response does not contain the results we wanted, that is, information about the tickets in Skylar One. Instead, the response contains:

- **HTTP/1.1 302 Found.** This status code indicates that *ticket* resources were found, but our request was missing required filtering and options.
- **Location: /ticket?limit=100.** This is a redirect header. Most browsers would automatically redirect our request to this URI. However, cURL requires an additional option to use redirects.
- **"X-EM7-status-message: ticket index requires a limit"** and **"X-EM7-status-code: FOUND"**. Human-readable status messages provided by the API. These messages indicate that our API does include ticket resources and that our HTTP request was missing the "limit" option.
- **"searchspec".** The response includes a list of searchspec options. These options allow us to filter the items (in this case, tickets) that are included in a response.
- **"total_matched":"102", "total_returned":0, "result_set":[].** This line indicates that the request could have returned 102 tickets, but that our request returned zero tickets.

Let's run the command again with the correct URI that contains the required option. To do this, enter the following at the command line:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
'https://192.168.10.205/api/ticket?limit=100'
```

The response looks like this:

```
* About to connect() to 192.168.10.205 port 443 (#0)

* Trying 192.168.10.205... connected

* Connected to 192.168.10.205 (192.168.10.205) port 443 (#0)
```

```
* Server auth using Basic with user 'em7admin'

> GET /api/ticket?limit=100 HTTP/1.1

> Authorization: Basic ZW03YWRtaW46ZW03YWRtaW4=

> User-Agent: curl/7.16.3 (powerpc-apple-darwin9.0) libcurl/7.16.3
OpenSSL/0.9.71 zlib/1.2.3

> Host: 192.168.10.205

> Accept: */*

> X-em7-beautify-response:1

>

< HTTP/1.1 200 OK

< Date: Wed, 25 Aug 2010 15:49:40 GMT

< Server: Apache

< X-EM7-Implemented-methods: GET,POST

< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow

< Transfer-Encoding: chunked

< Content-Type: application/json

<

{

"searchspec":{

"fields":{

"data":[

"class",

"severity",

"status",
```

```
"source",
"date_create",
"date_update",
"assigned_to",
"resolution",
"cause",
"escalation",
"chargeback",
"date_close",
"auto_close",
"organization",
"description",
"opened_by",
"updated_by",
"closed_by",
"ticket_queue",
"parent_ticket"
]
},
"options":{
  "extended_fetch":{
    "type":"boolean",
    "description":"Fetch entire resource if 1 (true), or resource link only if
0 (false)",
```

```
"default":"0"

},

"hide_filterinfo":{

"type":"boolean",

"description":"Suppress filterspec and current filter info if 1 (true)",

"default":"0"

},

"limit":{

"type":"int",

"description":"Number of records to retrieve",

"default":"100"

},

"offset":{

"type":"int",

"description":"Specifies the index of the first returned resource within
the entire result set",

"default":"0"

}

},

"total_matched":"102",

"total_returned":100,

"result_set":[

{
```



```
"URI":"\\api\\ticket\\1",

"description":"TICKET FOR ORGANIZATION: Device not responding to critical
ping "

},

{

"URI":"\\api\\ticket\\2",

"description":"TICKET FOR ORGANIZATION: Connection refused to port:
Timeout while requesting http:\\\\www.google.com"

},

[.... REMOVED TICKETS 3-98 FROM response, FOR BREVITY]

{

"URI":"\\api\\ticket\\99",

"description":"Rollback Configuration on Device CustB_2821-1.cisco.com"

},

{

"URI":"\\api\\ticket\\100",

"description":"Physical Memory usage has exceeded threshold: (80%)
currently (99%)"

}

]

}

* Connection #0 to host 192.168.10.205 left intact

* Closing connection #0
```

Notice that the response includes:

- **HTTP/1.1 200 OK.** Status code that indicates that our HTTP request was successful.
- **An entry for each of the first 100 tickets found.** The response includes basic information about the first 100 tickets found (as specified in the "limit" option). For each found ticket, The response includes:

- URI of the ticket, which includes the ticket ID.

NOTE: Our response is in JSON format. Notice that the URI for the ticket includes escaped forward slash characters ("\/").

- Description text from the ticket.
- To retrieve all information about a ticket, you can use the **extended_fetch** option. This is described in the following section.

Viewing a List of Tickets and Ticket Details

We can use the HTTP GET method and the **extended_fetch** option to retrieve all information about each returned ticket. If we append "&extended_fetch=1" to our URI, we can retrieve all information about the specified tickets. To do so, we enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
'https://192.168.10.205/api/ticket?limit=100&extended_fetch=1'
```

For each returned ticket, The response will include something like the following structure:

```
{
  "class": "1",
  "severity": "2",
  "status": "0",
  "source": "43",
  "date_create": "2010-01-18 20:12:06",
  "date_update": "2010-01-18 20:12:06",
  "assigned_to": "\/api\/account\/0",
  "resolution": "0",
```

```
"cause":"0",
"escalation":"0",
"chargeback":"0",
"date_close":"0000-00-00 00:00:00",
"auto_close":"0",
"custom_fields":{
},
"organization":"\\api\\organization\\0",
"description":"Rollback Configuration On Device CustB_2821-1.cisco.com",
"opened_by":"\\api\\account\\1",
"updated_by":"\\api\\account\\1",
"closed_by":"\\api\\account\\1",
"ticket_queue":"\\api\\ticket_queue\\8",
"parent_ticket":"\\api\\ticket\\0",
"aligned_resource":null,
"notes":{
  "URI":"\\api\\ticket\\99\\note\\/?hide_filterinfo=1&limit=1000",
  "description":"Notes"
}
}
```

Notice that the response now includes information about all the ticketing fields.

Also notice that some fields, like organization, include URIs. The URI is a link to a different resource (for example, an **organization** resource).

NOTE: Our response is in JSON format. Notice that these URIs include escaped forward slash characters ("V").

Filtering a List of Tickets

We can use the fields listed in `searchspec` to filter the list of tickets that will appear in the response. For the **ticket** resource, the ***searchspec*** includes:

- class
- severity
- status
- source
- date_create
- date_update
- assigned_to
- resolution
- cause
- escalation
- chargeback
- date_close
- auto_close
- organization
- description
- opened_by
- updated_by
- closed_by
- ticket_queue
- parent_ticket

In our example, we'll filter the list of tickets by severity.

- If we wanted to request only tickets with a severity of "major", we would append "&filter.severity=3" to the URI for our request. To view tickets of a specific severity, use the format:
 - "&filter.severity=number_of_severity"
- If we wanted to view all tickets with a severity of major or higher, we would append "&filter.severity.min=3" to the URI for our request. You can use the following operators in a filter clause:

- .not (not equal to)
- .min (greater than or equal to)
- .max (less than or equal to)
- .contains (string comparison)
- .in (is in a list)

To request all tickets with a severity of "major", enter the following at the command line:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/ticket?limit=100&extended_
fetch=1&filter.severity=3"
```

The response contains all ticket information for all tickets with a severity of "major".

To request all tickets with a severity equal to or greater than major (major and critical), enter the following at the command line:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/ticket?limit=100&extended_
fetch=1&filter.severity.min=3"
```

The response contains all ticket information for all tickets with a severity of "major" or "critical".

Retrieving Information about a Specific Ticket

We can use the HTTP GET method and the URI for a specific ticket to request information about only that specific ticket.

NOTE: When you include the URI for a specific ticket, the results automatically include all the information for the ticket. If you include the URI for a specific ticket, you do not need to include "&extended_fetch=1"

For example, if we wanted to request information about ticket 99, we could enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/ticket/99"
```

The response would look like this:

```
* About to connect() to 192.168.10.205 port 443 (#0)
```

```
* Trying 192.168.10.205... connected

* Connected to 192.168.10.205 (192.168.10.205) port 443 (#0)

* Server auth using Basic with user 'em7admin'

> GET /ticket/99 HTTP/1.1

> Authorization: Basic ZW03YWRTaW46ZW03YWRTaW4=

> User-Agent: curl/7.16.3 (powerpc-apple-darwin9.0) libcurl/7.16.3
OpenSSL/0.9.7l zlib/1.2.3

> Host: 192.168.10.205

> Accept: */*

> X-em7-beautify-response:1

>

< HTTP/1.1 200 OK

< Date: Wed, 25 Aug 2010 15:51:40 GMT

< Server: Apache

< X-EM7-Implemented-methods: GET,PUT,POST

< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow

< X-EM7-status-message: Ticket tid:99 loaded successfully

< X-EM7-status-code: OK

< Content-Length: 812

< Content-Type: application/json

<

{

"class": "1",

"severity": "2",
```

```
"status":"0",
"source":"43",
"date_create":"1263845526",
"date_update":"1263845526",
"assigned_to":"\\api\\account\\0",
"resolution":"0",
"cause":"0",
"escalation":"0",
"chargeback":"0",
"date_close":"0",
"auto_close":"0",
"organization":"\\api\\organization\\0",
"description":"Rollback Configuration On Device CustB_2821-1.cisco.com",
"opened_by":"\\api\\account\\1",
"updated_by":"\\api\\account\\1",
"closed_by":"\\api\\account\\1",
"ticket_queue":"\\api\\ticket_queue\\8",
"parent_ticket":"\\api\\ticket\\0",
"aligned_resource":null,
"custom_fields":{
},
"notes":{
"URI":"\\api\\ticket\\99\\note\\?hide_filterinfo=1&limit=1000",
"description":"Notes"
```

```
}  
  
}  
  
* Connection #0 to host 192.168.10.205 left intact  
  
* Closing connection #0
```

- Notice the HTTP status message and the ScienceLogic status messages.
- The response includes all the details about the specified ticket.

Updating a Ticket

The easiest way to update a ticket is to:

1. Use an HTTP GET request to capture the ticket's current values and store them in a file.
2. Edit that captured file.
3. Use an HTTP POST method to update the ticket with the contents of the edited file.

In this section, we'll update some values for ticket 99.

Capture Ticket Information in a File

To update a ticket, first we will capture the information from ticket 99 and store it in a file. To do this, enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'  
"https://192.168.10.205/api/ticket/99" > ticket99.json
```

We have now captured the information from ticket 99 and stored it in the file *ticket99.json*. The file looks like this:

```
{  
  
  "class": "1",  
  
  "severity": "2",  
  
  "status": "0",  
  
  "source": "43",  
  
  "date_create": "1263845526",  
  
  "date_update": "1263845526",
```



```
"assigned_to":"\\api\\account\\0",
"resolution":"0",
"cause":"0",
"escalation":"0",
"chargeback":"0",
"date_close":"0",
"auto_close":"0",
"organization":"\\api\\organization\\0",
"description":"Rollback Configuration On Device CustB_2821-1.cisco.com",
"opened_by":"\\api\\account\\1",
"updated_by":"\\api\\account\\1",
"closed_by":"\\api\\account\\1",
"ticket_queue":"\\api\\ticket_queue\\8",
"parent_ticket":"\\api\\ticket\\0",
"aligned_resource":null,
"custom_fields":{
},
"notes":{
"URI":"\\api\\ticket\\99\\note\\?hide_filterinfo=1&limit=1000",
"description":"Notes"
}
}
```

Edit the Captured File

To update the ticket, we'll edit one or more values in the file ticket99.json. Let's change the severity (from "2" (minor) to "4" (critical)) and status (from "0" (open) to "1" (working)) of the ticket and then save our changes to the file.

NOTE: Do not make changes to the value for *notes*. This is a sub-resource, which are explained later in this example.

```
{  
  
  "class": "1",  
  
  "severity": "4",  
  
  "status": "1",  
  
  "source": "43",  
  
  "date_create": "1263845526",  
  
  "date_update": "1263845526",  
  
  "assigned_to": "\/api\/account\/0",  
  
  "resolution": "0",  
  
  "cause": "0",  
  
  "escalation": "0",  
  
  "chargeback": "0",  
  
  "date_close": "0",  
  
  "auto_close": "0",  
  
  "organization": "\/api\/organization\/0",  
  
  "description": "Rollback Configuration On Device CustB_2821-1.cisco.com",  
  
  "opened_by": "\/api\/account\/1",  
  
  "updated_by": "\/api\/account\/1",  
  
}
```

```

"closed_by":"\\api\\account\\1",

"ticket_queue":"\\api\\ticket_queue\\8",

"parent_ticket":"\\api\\ticket\\0",

"aligned_resource":null,

"custom_fields":{

},

"notes":{

"URI":"\\api\\ticket\\99\\note\\/?hide_filterinfo=1&limit=1000",

"description":"Notes"

}

}

```

Use HTTP POST to Update the Ticket with the Edited File

We'll now use an HTTP POST method to update the ticket with the contents of the file *ticket99.json*. To do this, enter the following at the command line:

```

curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/ticket/99" -H 'content-type:application/json'
--data-binary @ticket99.json

```

In addition to the optional "beautify response" header and the URI for the ticket, you must specify:

- **'content-type:application/json'**. So the API knows that the incoming data is in JSON format.
- **--data-binary @file_name.json**. Specifies that HTTP POST should transmit the data exactly as is, with no extra processing. The @ symbol tells cURL that the data is stored in a file.

The response should look like the following:

```

* About to connect() to 192.168.10.205 port 443 (#0)

* Trying 192.168.10.205... connected

* Connected to 192.168.10.205 (192.168.10.205) port 443 (#0)

* Server auth using Basic with user 'em7admin'

> POST /ticket/99 HTTP/1.1

```

```
> Authorization: Basic ZW03YWRTaW46ZW03YWRTaW4=

> User-Agent: curl/7.16.3 (powerpc-apple-darwin9.0) libcurl/7.16.3
OpenSSL/0.9.71 zlib/1.2.3

> Host: 192.168.10.205

> Accept: */*

> X-em7-beautify-response:1

> content-type:application/json

> Content-Length: 722

>

< HTTP/1.1 200 OK

< Date: Wed, 25 Aug 2010 15:53:40 GMT

< Server: Apache

< X-EM7-Implemented-methods: GET,PUT,POST

< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow

< X-EM7-status-message: Ticket tid:99 updated.

< X-EM7-status-code: OK

< Content-Length: 812

< Content-Type: application/json

<

{

  "class":"1",

  "severity":"4",

  "status":"1",

  "source":"43",
```

```
"date_create":"1263845526",
"date_update":"1263845526",
"assigned_to":"\\api\\account\\0",
"resolution":"0",
"cause":"0",
"escalation":"0",
"chargeback":"0",
"date_close":"0",
"auto_close":"0",
"organization":"\\api\\organization\\0",
"description":"Rollback Configuration On Device CustB_2821-1.cisco.com",
"opened_by":"\\api\\account\\1",
"updated_by":"\\api\\account\\1",
"closed_by":"\\api\\account\\1",
"ticket_queue":"\\api\\ticket_queue\\8",
"parent_ticket":"\\api\\ticket\\0",
"aligned_resource":null,
"custom_fields":{
},
"notes":{
  "URI":"\\api\\ticket\\99\\note\\?hide_filterinfo=1&limit=1000",
  "description":"Notes"
}
}
```

```
* Connection #0 to host 192.168.10.205 left intact
```

```
* Closing connection #0
```

- Notice that the status codes and status messages specify that the ticket was updated and also specify the ticket ID. The response contains the ticket with the edits applied, so an additional GET request on the `/api/ticket/99` URI is not necessary to see the changes.
- If our file *ticket99.json* had included bad syntax, we would get:

```
HTTP/1.1 400 Bad Request
```

```
< X-EM7-status-message: Unable to decode JSON string.
```

```
< X-EM7-status-code: BAD_REQ
```

Sending Only Changes in the ticket99.json File

Our *ticket99.json* file included all the ticket information. However, we could have edited our file *ticket99.json* to include only changes to the ticket. That is, our file could contain only:

```
{  
  
  "severity": "4",  
  
  "status": "1"  
}
```

We could have sent this shortened file with an HTTP POST method and had the same result.

Creating a New Ticket

We can use the HTTP POST method to create a new ticket. To create a ticket, we must perform the following steps:

1. Capture an existing ticket and store the information in a file. We will use this file as our template for creating a new ticket.
2. Determine the URI for a user account. This is the user that will appear in the *opened_by*, *assigned_to*, and *updated_by* fields.
3. Edit the captured file to create the new ticket.
4. Perform an HTTP POST method to create a new ticket from the edited file.

The following sections explain each step.

Capturing an Existing Ticket and Storing the Information in a File

First, we will request all the information from an existing ticket and store that information in a file. We will then use the file as a template for creating a new ticket.

To do this, enter the following at the command line:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/ticket/1" > new_ticket.json
```

We will use the file *new_ticket.json* as our template.

Determining the URI for a User Account

The fields *opened_by*, *assigned_to*, and *updated_by* require a reference to a user account. Because **account** is another resource in the API, the reference for each of these fields is a URI for a specific account.

First, let's request the index for the **account** resource. To do this, enter the following at the command line:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/account"
```

The response looks like this:

[REMOVED CONNECTION INFORMATION AND SOME HEADER INFORMATION FROM RESPONSE, FOR BREVITY]

< HTTP/1.1 302 Found

< Date: Wed, 25 Aug 2010 15:54:52 GMT

< Server: Apache

< X-Implemented-methods: GET,POST

< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow

< Location: /account?limit=100

< X-EM7-status-message: account index requires a limit

< X-EM7-status-code: FOUND

< Content-Length: 1828

```
< Content-Type: application/json
```

```
<
```

```
{
```

```
"searchspec":{
```

```
"fields":{
```

```
"data":[
```

```
"default_map_type",
```

```
"user",
```

```
"email",
```

```
"state",
```

```
"restrict_ip",
```

```
"admin",
```

```
"active",
```

```
"create_date",
```

```
"edit_date",
```

```
"timezone",
```

```
"default_map",
```

```
"refresh",
```

```
"barred",
```

```
"page_results",
```

```
"event_severity",
```

```
"ldap",
```

```
"console_height",
```

```
"date_format",
```


"iflabel_pref",

"all_orgs",

"contact_fname",

"contact_lname",

"title",

"dept",

"phone",

"fax",

"cell",

"pager",

"email_2",

"address",

"office",

"city",

"zip",

"country",

"billing_id",

"crm_id",

"tollfree",

"email_3",

"im",

"im_type",

"role",

"critical",

```
"notes",
"verification_question",
"verification_answer",
"organization",
"theme",
"created_by",
"updated_by",
"user_policy"
]
},
"options":{
"extended_fetch":{
"type":"boolean",
"description":"Fetch entire resource if 1 (true), or resource link only if
0 (false)",
"default":"0"
},
"hide_filterinfo":{
"type":"boolean",
"description":"Suppress filterspec and current filter info if 1 (true)",
"default":"0"
},
"limit":{
"type":"int",
```

```

"description":"Number of records to retrieve",

"default":"100"

},

"offset":{

"type":"int",

"description":"Specifies the index of the first returned resource within
the entire result set",

"default":"0"

}

},

"total_matched":"26",

"total_returned":0,

"result_set":[

]

}

* Connection #0 to host 192.168.10.205 left intact

* Closing connection #0

```

The response tells us that:

- The account URI requires the limit option (like the previous example for the ticket resource).
- We can filter accounts by user-name.

We can now try to find the URI for our current user ID, em7admin. To do this, enter the following at the command prompt:

```

curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/account?limit=100&filter.user=em7admin"

```

The response will look like this:

```
[...REMOVED CONNECTION INFORMATION, HEADER INFORMATION, and  
SEARCHSPEC INFORMATION FOR BREVITY]
```

```
},
```

```
"total_matched":"1",
```

```
"total_returned":1,
```

```
"result_set":[
```

```
{
```

```
"URI":"\\/api\\/account\\/1",
```

```
"description":"em7admin"
```

```
}
```

```
]
```

```
}
```

```
* Connection #0 to host 192.168.10.205 left intact
```

```
* Closing connection #0
```

- We now know that the URI for the user "em7admin" is "/api/account/1".

NOTE: Our response is in JSON format. Notice that the URI for the account includes escaped forward slash characters ("V").

Editing the Captured File

We'll edit our captured file like this:

```
{
```

```
"class":"1",
```

```
"severity":"3",
```

```
"status":"1",
```

```
"source":"43",
```

```
"assigned_to":"\\api\\account\\1",
"resolution":"0",
"cause":"0",
"escalation":"0",
"chargeback":"0",
"date_close":"0",
"auto_close":"0",
"organization":"\\api\\organization\\0",
"description":"TICKET FOR ORGANIZATION: System | ID: 0",
"opened_by":"\\api\\account\\1",
"updated_by":"\\api\\account\\1",
"closed_by":"\\api\\account\\0",
"ticket_queue":"\\api\\ticket_queue\\7",
"parent_ticket":"\\ticket\\0",
"aligned_resource":null,
"custom_fields":{
},
"notes":{
"URI":"\\api\\ticket\\1\\note\\?hide_filterinfo=1&limit=1000",
"description":"Notes"
},
"logs":{
"URI":"\\api\\ticket\\1\\log\\?hide_filterinfo=1&limit=1000",
"description":"Logs"
```

```
}
```

```
}
```

- We changed the ***assigned_to***, ***opened_by***, and ***updated_by*** field to the URI for the user "em7admin". Because the file is in JSON format, we must escape the forward slash characters (/). Notice that we did so when specifying the account URI.
- We removed the entire line that contains ***"date_create"***. The API will automatically insert the current date and time in the new ticket.
- We removed the entire line that contains ***"date_update"***. The API will automatically insert the current date and time in the new ticket.
- We set the ***organization*** field to reference the System organization (URI is "/api/organization/0").
- We accepted the previous ticket's values for all other fields. However, you can edit these fields as you want. To determine a URI value for a field, do an HTTP GET request for the referenced resource (account, organization, ticket, ticket_queue).
- We left the entries for "notes" and "logs" sub-resources. The API ignores these fields and replaces them with empty fields that reference the new ticket's URI.

Using the Edited File to Create a New Ticket

To use the file new_ticket.json to create a new ticket, enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/ticket" -H 'content-type:application/json' --data-
binary @new_ticket.json
```

- Notice that unlike when we updated the ticket, in this example we POST to the URI for the general index for the **ticket** resource instead of POSTing to a URI for a ticket ID. This is because we do not yet have a ticket ID.
- Like when we updated a ticket, we include the following in the POST:
 - ***'content-type:application/json'***. So the API knows that the incoming data is in JSON format.
 - ***--data-binary @file_name.json***. Specifies that HTTP POST should transmit the data exactly as is, with no extra processing. The @ symbol tells cURL that the data is stored in a file.

You should get a response that looks something like this:

```
[...REMOVED CONNECTION INFORMATION AND SOME HEADER INFORMATION FROM RESPO
NSE, FOR BREVITY]
```

```
< HTTP/1.1 201 Created
```

```
< Date: Wed, 25 Aug 2010 15:54:52 GMT
```

```
< Server: Apache

< X-Implemented-methods: GET,PUT,POST

< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow

< Location: /ticket/279

< X-EM7-status-message: Ticket tid:279 added successfully

< X-EM7-status-code: CREATED

< Content-Length: 788

< Content-Type: application/json

<

{

  "class":"1",

  "severity":"3",

  "status":"1",

  "source":"43",

  "date_create":"1260402605",

  "date_update":"1260402605",

  "assigned_to":"\\account\\1",

  "resolution":"0",

  "cause":"0",

  "escalation":"0",

  "chargeback":"0",

  "date_close":"0",

  "auto_close":"0",

  "organization":"\\api\\organization\\0",
```

```

"description":"TICKET FOR ORGANIZATION: System | ID: 0",
"opened_by":"\\api\\account\\1",
"updated_by":"\\api\\account\\1",
"closed_by":"\\api\\account\\0",
"ticket_queue":"\\api\\ticket_queue\\7",
"parent_ticket":"\\api\\ticket\\0",
"aligned_resource":null,
"custom_fields":{
},
"notes":{
"URI":"\\api\\ticket\\note\\?hide_filterinfo=1&limit=1000",
"description":"Notes"
}
}

* Connection #0 to host 192.168.10.205 left intact

* Closing connection #0

```

- Notice that the status codes and status messages specify that the ticket was created successfully and also specify the ticket ID.
- If our file *new_ticket.json* had included bad syntax, we would get:

```

HTTP/1.1 400 Bad Request

< X-EM7-status-message: Unable to decode JSON string.

< X-EM7-status-code: BAD_REQ

...

```

- Notice that the API automatically inserted the current time (in UNIX timestamp format) for the "date_"

created" and "date_updated" fields.

- Notice that the API automatically inserted an appropriate URI for the "notes" sub-resource.

Viewing Notes for a Ticket

When you request information about a ticket, the response includes a **sub-resource: notes**. Sub-resources are always associated with their parent resource. Sub-resources have their own URI, appended to that of their parent resource. In our examples, **notes** is a sub-resource of a **ticket** resource.

We could look at the response from the ticket we just created (ticket 279). In the response, the reference to the notes a sub-resource looks like this:

```
"notes":{  
  
  "URI":"\\/ticket\\/note\\/?hide_filterinfo=1&limit=1000",  
  
  "description":"Notes"  
  
}
```

To view all the notes for the ticket we just created (ticket 279), enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'  
"https://192.168.10.205/api/ticket/279/note"
```

Because we have not yet added a note to this ticket, the response looks like this:

```
[.... REMOVED CONNECTION INFORMATION, HEADER INFORMATION,  
AND SEARCHSPEC INFORMATION FROM response, FOR BREVITY]
```

```
"total_matched":"0",
```

```
"total_returned":0,
```

```
"result_set":[
```

```
]
```

```
}
```

```
* Connection #0 to host 192.168.10.205 left intact
```

```
* Closing connection #0
```

Suppose we know that ticket ID 97 includes two notes. Let's request all the notes in this ticket. To do this, enter the following at the command line:

```
curl -v -H 'X-em7-beautify-response:1'
"https://192.168.10.205/api/ticket/97/note"
```

The response would look like this:

```
[.... REMOVED CONNECTION INFORMATION, HEADER INFORMATION,
AND SEARCHSPEC INFORMATION FROM response, FOR BREVITY]

"total_matched":"2",

"total_returned":2,

"result_set":[

{

"URI":"\\api\\ticket\\97\\note\\96",

"description":"Someone or some event altered the configuration on this
device.&nbsp; Roll back configuration to last-known-good.<br \\/>\\r\\n<br
\\/>\\r\\nEvent occured on device CustB_2821-1.cisco.com.<br \\/>\\r\\n<br
\\/>\\r\\nSee detail of event at
http:\\\\ap.server.url\\\\em7\\\\index.em7?exec=events&q_type=aid&q_
arg=17710&q_sev=1&q_sort=0&q_oper=0.<br \\/>\\r\\n<br \\/>\\r\\n<br \\/>\\r\\n<br
\\/>\\r\\n<br \\/>"

},

{

"URI":"\\api\\ticket\\97\\note\\270",

"description":"For security, immediately performed rollback.<br \\/>\\r\\n<br
\\/>\\r\\nCurrently analyzing logs to determine where change came from.<br
\\/>\\r\\n<br \\/>"

}

]

}

* Connection #0 to host 192.168.10.205 left intact

* Closing connection #0
```

In the response:

- We see that there are two notes in ticket 97: note 96 and note 270.
- We can view the text included in each note.

Now let's request a specific note. Using ticket 97 and our results above, we can request information about note 96 in ticket 97. To do this, enter the following at the command line:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'  
"https://192.168.10.205/api/ticket/97/note/96"
```

The response would look like this:

```
About to connect() to 192.168.10.205 port 443 (#0)
```

```
[.... REMOVED CONNECTION INFORMATION AND SOME HEADER INFORMATION  
FROM RESPONSE, FOR BREVITY]
```

```
< HTTP/1.1 200 OK
```

```
< Date: Wed, 25 Aug 2010 15:59:52 GMT
```

```
< Server: Apache
```

```
< X-Implemented-methods: GET,PUT,POST
```

```
< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow
```

```
< X-EM7-status-message: Note id:96 loaded successfully
```

```
< X-EM7-status-code: OK
```

```
< Content-Length: 475
```

```
< Content-Type: application/json
```

```
<
```

```
{
```

```
  "note_text": "Someone or some event altered the configuration on this  
device.&nbsp; Roll back configuration to last-known-good.<br \/\>\r\n<br  
\/\>\r\nEvent occured on device CustB_2821-1.cisco.com.<br \/\>\r\n<br  
\/\>\r\nSee detail of event at  
http:\\\\ap.server.url\\\\em7\\index.em7?exec=events&q_type=aid&q_  
arg=17710&q_sev=1&q_sort=0&q_oper=0.<br \/\>\r\n<br \/\>\r\n<br \/\>\r\n<br  
\/\>\r\n<br \/\>",
```

```
  "edited_by": "\\api\\account\\1",
```

```
  "date_edit": "1263845526",
```

```
  "ip": "192.168.10.206"
```

```
  "hidden": "0",
```

```
  "mime_type": "text\\html",
```

```
"media": {  
  
  "URI": "\/api\/ticket\/97\/note\/96\/media",  
  
  "description": "Associated Note Media"  
  
}  
  
}  
  
* Connection #0 to host 192.168.10.205 left intact  
  
* Closing connection #0
```

The response contains the following:

- HTTP code 200 OK. The API was able to successfully find note 96 within ticket 97. If note 96 did not exist, we would see the following:

```
HTTP/1.1 404 Not Found  
  
X-EM7-status-message: Note id:96 is not a valid note for ticket tid:97  
  
X-EM7-status-code: NOT_FOUND  
  

```

- Our note contains the following fields:
 - note_text
 - edited_by
 - date_edit
 - ip
 - hidden
 - mime_type

Adding a Note to a Ticket

Now let's try adding a note to an existing ticket. To do this:

We can use the HTTP POST method to add a note to an existing ticket. We will add a note to the ticket we created earlier, ticket 279. To add a note to a ticket, we must perform the following steps:

1. Request an existing note and store the information in a file. We will use this file as our template for creating a new note.
2. Edit the captured file.
3. Execute an HTTP POST method to create a new note from the edited file.

The following sections explain each step.

Capturing an Existing Note and Storing the Information in a File

To add a note to a ticket, first we will request the information from note 96 in ticket ID 97 and store it in a file. We will then use this file as a template. To do this, enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/ticket/97/note/96" > new_note.json
```

The information from the note will be stored in the file *new_note.json*. We will use this file as our template.

Editing the Captured File

We'll edit our file *new_note.json* like this:

```
{
  "hidden":"0",
  "note_text":"This is a test note from the API",
  "mime_type":"text/html",
  "edited_by":"\\/api\\/account\\/1"
}
```

- We removed the lines that contain *"date_edit"* and *"ip"*. The API will automatically insert the current date and time and the source IP of the request in the new note.
- We removed the *"media"* section.
- In the *mime_type* field, we accepted the value from the previous ticket (text/html).
- In the *hidden* field, we accepted the value from the previous ticket ("0", zero).
 - When "hidden" is set to "0" (zero), the note is not cloaked.
 - When "hidden" is set to "1" (one), the note is cloaked.

- We changed the value of the ***note_text*** field to "This is a test note from the API".
- We changed the ***edited_by*** field to the URI for the user "em7admin" (/api/account/1). Because the file is in JSON format, we must escape the forward slash characters (/). Notice that we did so when specifying the account URI.

Creating a New Note Using the Edited File

To use the file ***new_note.json*** to create a new note in ticket 279, enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/ticket/279/note" -H 'content-
type:application/json' --data-binary @new_note.json
```

- Notice that we POST to the URI for the index for the ***note*** sub-resource for this ticket, instead of to a specific note ID. This is because we do not yet have a note ID.
- We include the following in the POST:
 - ***'content-type:application/json'***. So the API knows that the incoming data is in JSON format.
 - ***--data-binary @file_name.json***. Specifies that HTTP POST should transmit the data exactly as is, with no extra processing. The @ symbol tells cURL that the data is stored in a file.

The response should look like the following:

```
[.... REMOVED CONNECTION INFORMATION and SOME HEADER INFORMATION
FROM RESPONSE, FOR BREVITY]
```

```
< HTTP/1.1 201 Created
```

```
< Date: Wed, 25 Aug 2010 16:01:49 GMT
```

```
< Server: Apache
```

```
< X-Implemented-methods: GET,PUT,POST
```

```
< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow
```

```
< Location: /api/ticket/279/note/273
```

```
< X-EM7-status-message: Note /ticket/279/note/273 added.
```

```
< X-EM7-status-code: CREATED
```

```
< Content-Length: 142
```

```
< Content-Type: application/json
```

```
<
{
  "date_edit": "1264525835",
  "hidden": "0",
  "note_text": "This is a test note from the API",
  "edited_by": "\\api\\account\\1"
}

* Connection #0 to host 192.168.10.205 left intact

* Closing connection #0
```

- Notice that the status codes and status messages specify that the note was created successfully and also specify the note ID (note 273).
- If our file *new_ticket.json* had included bad syntax, we would get:

```
HTTP/1.1 400 Bad Request

< X-EM7-status-message: Unable to decode JSON string.

< X-EM7-status-code: BAD_REQ

...
```

- Notice that the API automatically inserted the current date and time (in UNIX timestamp format) in the *date_edited* field.

Viewing the Attachments for a Ticket

In a ticket, each note can include one or more attachments. Each ticket note has a `/media` sub-resource that can be used to search and view the attachments associated with that ticket note.

For example, to view the attachments for ticket 2058, at note 11, we could enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
'https://192.168.10.205/api/ticket/2058/note/11/media?limit=100'
```

The response would look like this:

*[.... REMOVED CONNECTION INFORMATION AND SOME HEADER INFORMATION
FROM RESPONSE, FOR BREVITY]*

< HTTP/1.1 200 OK

< Date: Wed, 25 Aug 2010 16:03:11 GMT

< Server: Apache

< X-Implemented-methods: GET

< X-Powered-By: ScienceLogic, LLC - EM7 API/Skylar One PowerFlow

< **X-EM7-status-message: Fileindex found with 1 resources.**

< **X-EM7-status-code: OK**

< **Content-Disposition: inline; filename="test_attachment.rtf"**

< Content-Length: 357

< Content-Type: application/rtf

<

{

"searchspec": {

"fields": [

"creation_date",

"file_length",

"is_attachment",

"is_complete",

"mime_type",

"modified_date",

"total_size",

"user_owner"

```
],  
  
"options": {  
  "hide_filterinfo": {  
    "type": "boolean",  
    "description": "Suppress filterspec and current filter info if 1  
    (true)",  
    "default": "0"  
  },  
  
  "limit": {  
    "type": "int",  
    "description": "Number of records to retrieve",  
    "default": "100"  
  },  
  
  "offset": {  
    "type": "int",  
    "description": "Specifies the index of the first returned resource  
    within the entire result set",  
    "default": "0"  
  },  
  
  "extended_fetch": {  
    "type": "boolean",  
    "description": "Fetch entire resource if 1 (true), or resource  
    link only if 0 (false)",  
    "default": "0"  
  },  
}
```

```

"link_disp_field": {
  "type": "enum",
  "description": "When not using extended_fetch, this determines
which field is used for the \"description\" of the resource link",
  "default": "mime_type",
  "values": [
    "is_attachment",
    "mime_type",
    "is_complete",
    "user_owner",
    "total_size",
    "file_length",
    "creation_date",
    "modified_date"
  ]
}
},
"total_matched": 1,
"total_returned": 1,
"result_set": [
  {
    "URI": "\\api\\ticket\\2058\\note\\11\\media\\Penguins.jpg\\info",
    "description": "image\\jpeg"
  }
]

```

```

    }
]

* Connection #0 to host 192.168.10.205 left intact

* Closing connection #0

```

In the response, notice that:

- The status codes and messages specify that one attachment was found.
- If no attachments were found, the response would include:

```
X-EM7-status-message: Fileindex found with 0 resources.
```

- The result set includes an entry for each attachment.
- The entry for each attachment includes the URI that can be used to request detailed information about the attachment.

Now let's request a specific attachment. Using ticket 2058, note 11 and our results above, we can request information about the "Penguins.jpg" attachment. To do this, enter the following at the command line:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/ticket/2058/note/11/media/Penguins.jpg/info"
```

The response would look like this:

```
About to connect() to 192.168.10.205 port 443 (#0)

[.... REMOVED CONNECTION INFORMATION AND SOME HEADER INFORMATION
FROM RESPONSE, FOR BREVITY]

< HTTP/1.1 200 OK

< Date: Wed, 25 Aug 2010 15:59:52 GMT

< Server: Apache

< X-Implemented-methods: GET,POST,PUT,DELETE

< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow

< X-EM7-status-message: /ticket/2058/note/11/media/Penguins.jpg loaded
successfully

< X-EM7-status-code: OK

< Content-Length: 475

```

```
< Content-Type: application/json
<
{
  "mime_type": "image/jpeg",
  "is_complete": "1",
  "user_owner": "\\api\\account\\0",
  "total_size": 777835,
  "file_length": 777835,
  "creation_date": "1445379816",
  "modified_date": "1445379816",
  "is_attachment": "1",
  "chunks": [
    {
      "offset": 0,
      "length": 777835,
      "md5": "9d377b10ce778c4938b3c7e2c63a229a"
    }
  ],
  "data": {
    "URI": "\\api\\ticket\\2058\\note\\11\\media\\Penguins.jpg",
    "description": "File Contents"
  }
}

* Connection #0 to host 192.168.10.205 left intact
```

```
* Closing connection #0
```

The response contains the following:

- HTTP code 200 OK. The API was able to successfully find the Penguins.jpg attachment.
- The URI field in the data section specifies the download link for the file.

NOTE: For FIPS-compliant systems, the response will include a SHA hash, not an MD5 hash.

Adding an Attachment to a Ticket Note

You can add an attachment to an existing ticket note. To do this, we must use the HTTP PUT method instead of the HTTP POST method. The HTTP PUT method is used for explicitly adding or replacing (where HTTP POST is used for creating or updating).

The API will not allow you to add an attachment with an HTTP POST method. If you try, the response will look like this:

```
[.... REMOVED CONNECTION INFORMATION AND SOME HEADER INFORMATION  
FROM RESPONSE, FOR BREVITY]
```

```
< HTTP/1.1 405 Method Not Allowed
```

```
< Date: Wed, 25 Aug 2010 16:04:25 GMT
```

```
< Server: Apache
```

```
< X-Implemented-methods: GET,PUT,DELETE
```

```
< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow
```

```
< X-EM7-status-message: POST not allowed for note media. PUT an explicitly  
named new attachment or image
```

```
< X-EM7-status-code: BAD_METHOD
```

```
< Content-Length: 214
```

```
< Content-Type: application/json
```

```
<
```

```
{
```

```
"errors":[
```

```
{
  "errorcode":"BAD_METHOD",
  "message":"POST not allowed for ticket attachments. PUT an explicitly
named new attachment or image"
}
],
"messages":[
],
"resource_body":null
}

* Connection #0 to host 192.168.10.205 left intact

* Closing connection #0
```

To use HTTP PUT, we must include the "-T" option with the cURL command.

Suppose we want to add the image file "spidey.png" to ticket 97, note 96. We could enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/ticket/97/note/96/media/spidey.png" -H
"content-type:image/png" -T ./spidey.png
```

- **attachment/spidey.png**. Tells the API the filename to use when saving the attachment in the system.
- **-H "content-type:image/png"**. Tells the API that the attachment will be an image file of type png.
- **-T**. Tells cURL to perform an HTTP PUT.
- **./spidey.png**. Full pathname of the file to attach. "." means "current directory".
- Notice that unlike HTTP POST, the HTTP PUT method does not require the "--data-binary option" or the "@" characters before the filename.

NOTE: If an attachment has been prohibited in the **Ticket Attachment Blacklist** page (Registry > Ticketing > Attachment Blacklist), the API will not attach the file to the ticket note. The API will not allow you to attach files with a file extension that matches a blacklist entry.

Example

2

Using the Discovery Resource

Overview

Discovery is the process by which Skylar One retrieves data from the devices in a network and then adds and configures those devices. Skylar One runs discovery to perform the initial discovery of your network and then nightly to collect and update information about your network. You can also manually initiate discovery, for a single device or for a range of devices, at any time.

To start discovery, you must provide the discovery tool with one or more IP addresses and other information about how you want Skylar One to perform the discovery. You save the list of IP addresses and other information about how you want Skylar One to perform the discovery in a **discovery session**. When you execute the discovery session, Skylar One then finds all the devices and components in the range. For each discovered device or component, Skylar One gathers detailed data. This data is used throughout Skylar One.

This chapter will show you how to use the API to perform some basic tasks for managing discovery sessions.

This chapter covers the following topics:

Requirements	154
Getting Started	154
Connecting to the API	155
Viewing a List of Discovery Sessions	159
Viewing Details about All Discovery Sessions	166
Filtering the List of Discovery Sessions	168
Retrieving Information about a Specific Discovery Session	169

<i>Starting a Discovery Session</i>	<i>171</i>
<i>Viewing a List of All Active Discovery Sessions</i>	<i>173</i>
<i>Retrieving Information about a Specific Active Discovery-Session</i>	<i>175</i>
<i>Viewing Logs for a Discovery Session</i>	<i>176</i>
<i>Stopping a Currently Running Discovery-Session</i>	<i>179</i>
<i>Deleting a Discovery Session</i>	<i>181</i>

Requirements

- This chapter assumes that you have a working version of cURL installed and can run cURL from a command prompt. For information on cURL, see <http://curl.haxx.se/>.
- To connect to the API, **you must use HTTPS**. If you have not installed or configured a security certificate or if your appliance uses a self-signed certificate, you must use include the "-k" option each time you execute cURL. The "-k" option tells cURL to perform the HTTPS connection without checking the security certificate.
- Through the API, you can perform only actions for which you have permission in the system. To perform the tasks in this chapter, you must connect to the API using an account (username and password). The account must have Access Keys that grant the following:
 - Access the Discovery page
 - Schedule or execute a discovery session
- In order for the hooks you specify during the creation of a Discovery Session using the ScienceLogic API to be effective, you must also run the discovery through the ScienceLogic API. If the discovery is run from the user interface, the API will disregard the hooks.

Getting Started

- In the examples in this chapter, we will connect to the example Administration Portal with the IP address of 192.168.10.205. To run these examples on your system, you should replace this IP address with the base URI of the API on the appliance you are using.
- In the examples in this chapter, we will connect using the default account "em7admin" with the example password "examplepassword". To run these examples on your system, you should replace this username and password with a valid username and password for your system.
- In the examples in this chapter, we will execute each request at a shell prompt or command prompt. However, you can include these requests in a script or program.

CAUTION: The examples in this chapter use the custom-header option "***X-em7-beautify-response:1***". This header tells the API to include white-space in a response, to make it easier to read. However, this header can greatly increase the amount of time required to process a request. ***ScienceLogic recommends you use this header only when testing requests. ScienceLogic strongly discourages you from using this header in integration code.***

Connecting to the API

To connect to the API and view the root directory (with an HTTP GET request), enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api"
```

- ***curl -v***. Executes the cURL request. The -v option tells cURL to use verbose mode (displays all header information and all status and error messages). In the response, lines that start with ">" include header data returned by cURL. Lines that start with "<" include header data received by cURL.
- ***-H 'X-em7-beautify-response:1'***. The -H option tells cURL to include an additional header in the request. In this case, we're including a ScienceLogic custom header that tells the API to include white-space in the response.
- ***-u 'em7admin:examplepassword'***. The -u option tells cURL to authenticate as a specified user. In our example, we authenticated as the user "em7admin" with the password "examplepassword".
- ***"https://192.168.10.205/api"***. Connect to the specified URL. In our example, we connected to the API at 192.168.10.205.

The response will look like this (however, we've added line numbers for reference):

```
1) * About to connect() to 192.168.10.205 port 443 (#0)
2) * Trying 192.168.10.205... connected
3) * Connected to 192.168.10.205 (192.168.10.205) port 443 (#0)
4) * Server auth using Basic with user 'em7admin'
5) GET / HTTP/1.1
6) Authorization: Basic ZW03YWRTaW46ZW03YWRTaW4=
7) User-Agent: curl/7.16.3 (powerpc-apple-darwin9.0) libcurl/7.16.3
  OpenSSL/0.9.7l zlib/1.2.3
8) Host: 192.168.10.205
9) Accept: */*
10) X-em7-beautify-response:1
11) >
```

```
12) < HTTP/1.1 200 OK
13) < Date: Wed, 20 Jan 2010 23:03:46 GMT
14) < Server: Apache/2.2.9 (Unix) mod_ssl/2.2.9 OpenSSL/0.9.8e-fips-rhel5
15) < X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow
16) < Content-Length: 703
17) < Content-Type: application/json
18) <
19) {
20) "data":{
21) "account":{
22) "URI":"\\/account",
23) "description":"Get\\/Update\\/Add\\/Delete User Accounts"
24) },
25) "device":{
26) "URI":"\\/device?limit=100",
27) "description":"Get\\/Update\\/Add\\/Delete Devices and Get Collected
Data"
28) },
29) "discovery_session":{
30) "URI":"\\/discovery_session\\/",
31) "description":"Get\\/Update\\/Add\\/Delete Device Discovery Sessions"
32) },
33) "dynamic_app":{
34) "URI":"\\/dynamic_app\\/",
```

```

35) "description":"Get Dynamic Application Resources"
36) },
37) "organization":{
38) "URI":"\/organization",
39) "description":"Get\/Update\/Add\/Delete Organizations"
40) },
41) "ticket":{
42) "URI":"\/ticket?limit=100",
43) "description":"Get\/Update\/Add\/Delete Tickets"
44) },
45) "ticket_queue":{
46) "URI":"\/ticket_queue",
47) "description":"Get Ticket Queues"
48) }
49) }
50) }
51) Connection #0 to host 192.168.10.205 left intact
52) Closing connection #0

```

- Lines 1-4 show cURL trying to connect to and authenticate with the API.
- Lines 5-11 show the HTTP GET request we sent. The initial request performs a GET on the root directory of the API.
 - **accept: */***. Specifies that we will use the default accept header. The accept header tells the API how to format the response. The API can respond in XML or JSON. Because we didn't specify an accept header, the API will use the default format, which is JSON. If you want to view the response in XML, you can include the header option "-H 'Accept:application/xml'" in the cURL command.

- ***X-em7-beautify-response:1***. Tells the API to include white-space in the response, for easier reading.
- Line 12 shows the HTTP version and the HTTP status code for the response.
- Lines 12-18 show the header information for the response.
- Lines 19-52 display the response to the HTTP GET request on the root directory of the API.

The response for the HTTP GET request displays a list of resources. A **resource** is a functional area in Skylar One that you can access through the API.

You can interact with the following entities through the API:

- ***Accounts***
- ***Account Lockouts***
- ***Alerts***
- ***Appliances***
- ***Assets***
- ***Collector Groups***
- ***CBQoS Objects***
- ***Collection Labels***
- ***Credentials***
- ***Custom Attributes***
- ***Dashboards***
- ***Devices***
- ***Device Categories***
- ***Device Classes***
- ***Device Interfaces***
- ***Device Groups***
- ***Device Relationships***
- ***Device Templates***
- ***Discovery Sessions***
- ***Dynamic Applications***
- ***Events***
- ***Event Categories***
- ***External Contacts***
- ***File Uploads***
- ***Interfaces***
- ***Monitoring Policies***
- ***Organizations***
- ***Performance Data***

- *PowerPacks*
- *Product SKUs*
- *Schedules*
- *System Patches*
- *System Settings*
- *Tasks*
- *System Thresholds*
- *Themes*
- *Thresholds*
- *Tickets*
- *Ticket Categories*
- *Ticket Chargeback*
- *Ticket Logs*
- *Ticket Notes*
- *Ticket Queues*
- *Ticket States*
- *User Policies*
- *Vendors*

For each resource, the response displays the associated URI for accessing the resource and a description that lists the actions you can perform on the resource.

For our example, we'll be looking at the *discovery_session* resource. The entry for the **discovery_session** resource includes the URI of the **discovery_session** resource and includes the following description:

```

29) "discovery_session":{
30) "URI":"\\discovery_session\\",
31) "description":"Get\\Update\\Add\\Delete Device Discovery Sessions"
32) },

```

Viewing a List of Discovery Sessions

In the previous section, we used an HTTP GET request to retrieve information about the root directory of the API. Our response included a list of resources. We learned that we can request information about discovery sessions.

To access a resource, like *discovery_session*, we can use an HTTP GET and append a discovery session's URI to the URI of the root directory. To access the resource *discovery_session*, we could enter the following at the command line.

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/discovery_session"
```

The response looks like this:

```
...[REMOVED CONNECTION INFORMATION AND SOME HEADER INFORMATION FROM
RESPONSE, FOR BREVITY]
```

```
< HTTP/1.1 302 Found
```

```
< Date: Wed, 27 Jan 2010 16:32:05 GMT
```

```
< Server: Apache
```

```
< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow
```

```
< Location: /discovery_session?limit=100
```

```
< X-EM7-status-message: discovery_session index requires a limit
```

```
< X-EM7-status-code: FOUND
```

```
< Content-Length: 977
```

```
< Content-Type: application/json
```

```
<
```

```
{
```

```
"searchspec":{
```

```
"fields":{
```

```
"data":[
```

```
"dgid",
```

```
"date_edit",
```

```
"date_run",
```

```
"scan_ports",
```



```
"organization",
"collector_id",
"edited_by",
"discover_non_snmp"
]
},
"options":{
"extended_fetch":{
"type":"boolean",
"description":"Fetch entire resource if 1 (true), or resource link only if
0 (false)",
"default":"0"
},
"hide_filterinfo":{
"type":"boolean",
"description":"Suppress filterspec and current filter info if 1 (true)",
"default":"0"
},
"limit":{
"type":"int",
"description":"Number of records to retrieve",
"default":"100"
},
"offset":{
```

```

"type":"int",

"description":"Specifies the index of the first returned resource within
the entire result set",

"default":"0"

}

},

"total_matched":"14",

"total_returned":0,

"result_set":[

]

}

* Connection #0 to host 192.168.10.205 left intact

* Closing connection #0

```

The response does not contain the results we wanted, that is, information about the discovery sessions in Skylar One. Instead, the response contains:

- **HTTP/1.1 302 Found.** This status code indicates that *discovery_session* resources were found, but our request was missing required filtering and options.
- **Location: /discovery_session?limit=100.** This is a redirect header. Most browsers would automatically redirect our request to this URI. However, cURL requires an additional option to use redirects.
- **"X-EM7-status-message: discovery_session index requires a limit"** and **"X-EM7-status-code: FOUND"**. Human-readable status messages provided by the API. These messages indicate that our API does include *discovery_session* resources and that our HTTP request was missing the "limit" option.
- **"searchspec"**. The response include a list of searchspec options. These options allow use to filter content contained in the response.
- **"total_matched":"14", "total_returned":0, "result_set":[]**. This line indicates that the default "limit" option would have returned 14 discovery sessions, but that our request returned zero discovery sessions.

Let's run the command again with the correct URI that contains the required option. To do this, enter the following at the command line:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'  
"https://192.168.10.205/api/discovery_session?limit=100"
```

The response looks like this:

```
...[REMOVED CONNECTION INFORMATION AND SOME HEADER INFORMATION FROM  
RESPONSE, FOR BREVITY]
```

```
< HTTP/1.1 200 OK
```

```
< Date: Wed, 27 Jan 2010 16:36:20 GMT
```

```
< Server: Apache
```

```
< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow
```

```
< Content-Length: 2530
```

```
< Content-Type: application/json
```

```
<
```

```
{
```

```
"searchspec":{
```

```
"fields":{
```

```
"data":[
```

```
"dgid",
```

```
"date_edit",
```

```
"date_run",
```

```
"scan_ports",
```

```
"organization",
```

```
"collector_id",
```

```
"edited_by",
```

```
"discover_non_snmp"
```

```
]
```

```

},

"options":{

"extended_fetch":{

"type":"boolean",

"description":"Fetch entire resource if 1 (true), or resource link only if
0 (false)",

"default":"0"

},

"hide_filterinfo":{

"type":"boolean",

"description":"Suppress filterspec and current filter info if 1 (true)",

"default":"0"

},

"limit":{

"type":"int",

"description":"Number of records to retrieve",

"default":"100"

},

"offset":{

"type":"int",

"description":"Specifies the index of the first returned resource within
the entire result set",

"default":"0"

}

}

```

```

},
"total_matched":"14",
"total_returned":14,
"result_set":[
{
  "":{
    "URI":"\\discovery_session\\1",
    "description":"SNMP:1:2"
  }
},
{
  "":{
    "URI":"\\discovery_session\\2",
    "description":"21:22:23:25:80"
  }
},
[...REMOVED SESSIONS 3-13 FROM RESPONSE, FOR BREVITY]
{
  "":{
    "URI":"\\discovery_session\\14",
    "description":"21:22:23:25:80"
  }
},
{

```

```

"":{

"URI":"\\/discovery_session\\/15",

"description":"21:22:23:25:80"

}

}

]

}

* Connection #0 to host 192.168.10.205 left intact

* Closing connection #0

```

Notice that the response includes:

- **HTTP/1.1 200 OK.** Status code that indicates that our HTTP request was successful.
- **An entry for all of the discovery sessions found.** The response includes basic information about 14 discovery sessions, because only 14 sessions exist on our example system. Because we set the limit option to "100", the response could contain information about the first 100 discovery sessions. For each found discovery session, the response includes:
 - URI of the discovery session.

NOTE: Our response is in JSON format. Notice that the URI for the discovery session includes escaped forward slash characters ("V").

- Description of the discovery method. A list of values that were selected in the **Detection Method & Port** field (in the System > Manage > Classic Discovery or System > Manage > Discovery in the classic user interface page). The possible values are described in the following section, in the description of the **scan_ports** field.
- To request all information about all discovery sessions, you can use the **extended_fetch** option. This is described in the following section.

Viewing Details about All Discovery Sessions

We can use the HTTP GET request and the **extended_fetch** option to request all information about each returned discovery session. If we append "&extended_fetch=1" to our URI, we can request all information about the first 100 discovery sessions. To do so, we enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/discovery_session?limit=100&extended_fetch=1"
```

The response will return a list of the first 100 discovery sessions, with the following information for each session:

- **Discovery Session ID.** The unique numeric identifier, assigned to this session by Skylar One.

NOTE: The Discovery Session ID and the numbered list in the **Session Register** pane in the System > Manage > Classic Discovery or System > Manage > Discovery in the classic user interface page may not match. The **Session Register** pane in the System > Manage > Classic Discovery or System > Manage > Discovery in the classic user interface page is sorted by date and changes when a discovery session is edited.

- **dgid.** ID of the Device Group associated with this discovery session. If no device group is associated with this discovery session, this field will contain the value "NULL".
- **date_edit.** Date and time the discovery session was last edited.
- **date_run.** Date and time the discovery session was last executed.
- **scan_ports.** A list of values that were selected in the **Detection Method & Port** field (in the System > Manage > Classic Discovery or System > Manage > Discovery in the classic user interface page).
 - If in the **Detection Method & Port** field, a user selected the "Default" method, this list includes the default TCP ports that are used during discovery (21, 22, 23, 25, and 80).
 - If in the **Detection Method & Port** field, a user selected one or more TCP ports, the list includes those ports.
 - If in the **Detection Method & Port** field, a user selected "UDP 161", the list includes the string "SNMP".
- **ip_list.** The start IP and end IP for each IP range included in the discovery session.
- **credentials.** One or more credentials selected for this discovery session.
- **organization.** Link to the **organization** resource associated with the discovery session.
- **collector_id.** The appliance ID of the Data Collector associated with the discovery session.
- **edited_by.** Link to the **account** resource for the user who last edited this discovery session.
- **discover_non_snmp.** Specifies whether this session will discover devices that don't support SNMP. These devices are called "pingables" in Skylar One. "0" (zero) means do not include pingables; "1" (one) means include pingables.
- **logs.** Link to the logs sub-resource for this discovery session.

NOTE: Our response is in JSON format. Notice that the URIs for other resources include escaped forward slash characters ("\/").

Filtering the List of Discovery Sessions

We can use the fields listed in *searchspec* to filter the list of discovery sessions that will appear in the response. For the *discovery_session* resource, the *searchspec* includes:

- *dgid*. Selected Device Group.
- *date_edit*. Date and time the session was last edited.
- *date_run*. Date and time the session was last run.
- *scan_ports*. Value selected in the *Detection Method & Ports* field.
- *organization*. Organization associated with the discovery session.
- *collector_id*. Appliance ID of a single Data Collector (not a collector group). Currently, there is no way to query Appliance information through the API.
- *edited_by*. The user account that last edited the discovery session.
- *discover_non_snmp*. Specifies whether this session will discover devices that don't support SNMP. These devices are called "pingables" in Skylar One. "0" (zero) means do not include pingables; "1" (one) means include pingables.

If we wanted to view details about only discovery sessions that do not include pingables, we could enter the following at the command line:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/discovery_session?limit=100&extended_
fetch=1&filter.discover_non_snmp=0"
```

The response would display full details about the first 100 discovery sessions that do not discover pingables.

We can also use the following operators in the *searchspec*:

- .not (not equal to)
- .min (greater than or equal to)
- .max (less than or equal to)
- .contains (string comparison)
- .in (is in a list)

For example, if we wanted to view full details about the first 100 discovery sessions that are not associated with the Data Collector with the ID of 3, we could enter the following at the command line:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/discovery_session?limit=100&extended_
fetch=1&filter.collector_id.not=3"
```

Retrieving Information about a Specific Discovery Session

We can use the HTTP GET method and the URI for a specific discovery session to request information about that specific discovery session.

NOTE: When you include the URI for a specific discovery session, the response automatically includes all the information for the session. If you include the URI for a specific discovery session, you do not need to include "extended_fetch=1"

For example, if we wanted to request information about discovery session ID "1", we could enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/discovery_session/1"
```

The response would look like this:

```
...[REMOVED CONNECTION INFORMATION AND SOME HEADER INFORMATION FROM
RESPONSE, FOR BREVITY]
```

```
< HTTP/1.1 200 OK
```

```
< Date: Wed, 27 Jan 2010 19:16:49 GMT
```

```
< Server: Apache
```

```
< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow
```

```
< X-EM7-status-message: discovery_session seq:1 loaded successfully
```

```
< X-EM7-status-code: OK
```

```
< Content-Length: 625
```

```
< Content-Type: application/json
```

```
<
```

```
{
```

```
"dgid":null,
```

```
"date_edit":"1264540686",  
"date_run":"1264544101",  
"scan_ports":[  
  "SNMP",  
  "1",  
  "2"  
],  
"ip_lists":[  
  {  
    "start_ip":"192.168.9.1",  
    "end_ip":"192.168.9.100"  
  },  
  {  
    "start_ip":"192.168.10.200",  
    "end_ip":"192.168.10.203"  
  }  
],  
"credentials":[  
  "\\api\\credential\\snmp\\1",  
  "\\api\\credential\\snmp\\2",  
  "\\api\\credential\\db\\10"  
],  
"organization":"\\api\\organization\\0",  
"collector_id":"3",
```

```

"edited_by":"\\api\\account\\28",

"discover_non_snmp":"0",

"logs":{

"URI":"\\api\\discovery_session\\1\\log",

"description":"Discovery Session Logs"

}

}

* Connection #0 to host 192.168.10.205 left intact

* Closing connection #0

```

- Notice the HTTP status message and the ScienceLogic status messages.
- The response includes all the details about the specified discovery session
- If the discovery session did not exist (for example, if we supplied an incorrect ID), the response would include:

```

HTTP/1.1 404 Not Found

X-EM7-status-message: Unable to find discovery_session w/ id of '401'

X-EM7-info-message: Use /discovery_session search to find valid
discovery_session resources

```

Starting a Discovery Session

After the initial discovery, Skylar One automatically polls monitored devices and applications to retrieve new and updated data. Skylar One performs these updates at regular intervals. However, you can manually execute a discovery session at any time.

You can use the API to manually execute an existing discovery session. To do this, we use the URI of the existing discovery session, the HTTP POST method, and an additional resource called **discovery_session_active**.

The **discovery_session_active** resource allows you to execute a discovery session, view a list of currently active discovery sessions, and stop a currently active discovery session.

In our example, we'll POST the URI for a discovery session to the **discovery_session_active** resource. We'll use discovery session ID 1 as the discovery session.

To execute discovery session 1, enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/discovery_session_active" -H 'content-
type:application/em7-resource-uri' --data-binary "/api/discovery_
session/1"
```

- **"https://192.168.10.205/api/discovery_session_active"**. Notice that the address for the HTTP Post includes the **discovery_session_active** resource in the URI.
- **'content-type:application/em7-resource-uri'**. Tells the API that the incoming data is a resource URI.
- **--data-binary "/api/discovery_session/1"**. Specifies that HTTP POST should transmit use the URI exactly as is, with no extra processing. The URI of the discovery session must be surrounded by double-quotes.

The response looks like this:

```
[...REMOVED CONNECTION INFORMATION AND SOME HEADER INFORMATION FROM
RESPONSE, FOR BREVITY]
```

```
< HTTP/1.1 202 Accepted
```

```
< Date: Wed, 27 Jan 2010 19:36:44 GMT
```

```
< Server: Apache
```

```
< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow
```

```
< X-EM7-status-message: Discovery session queued for discovery
```

```
< X-EM7-status-code: ACCEPTED
```

```
< Content-Length: 1
```

```
< Content-Type: application/json
```

```
<
```

```
* Connection #0 to host 192.168.10.205 left intact
```

```
* Closing connection #0
```

- The HTTP status code is 202, because the action did not complete within the HTTP response time. This is because the discovery session is still running when the API generates the response.

- Notice the ScienceLogic status messages, which specify that the session has been queued for execution.
- If the discovery session is already running or is already queued, the response includes:

```
HTTP/1.1 400 Bad Request
```

```
X-EM7-status-message: /discovery_session/1 is already active
```

```
X-EM7-status-code: BAD_REQ
```

Viewing a List of All Active Discovery Sessions

You can use HTTP GET and the *discovery_session_active* resource index to request a list of all currently active discovery sessions.

To view a list of all currently active discovery sessions, enter the cURL command from the previous section, to start a discovery session:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/discovery_session_active" -H 'content-
type:application/em7-resource-uri' --data-binary "/api/discovery_
session/1"
```

Then immediately enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/discovery_session_active?limit=100"
```

The response looks like this:

```
* About to connect() to 192.168.10.205 port 443 (#0)
* Trying 192.168.10.205... connected
* Connected to 192.168.10.205 (192.168.10.205) port 443 (#0)
* Server auth using Basic with user 'em7admin'
> GET /discovery_session_active?limit=100 HTTP/1.1
> Authorization: Basic ZW03YWRTaW46ZW03YWRTaW4=
> User-Agent: curl/7.16.3 (powerpc-apple-darwin9.0) libcurl/7.16.3
OpenSSL/0.9.7l zlib/1.2.3
> Host: 192.168.10.205
> Accept: */*
> X-em7-beautify-response:1
>
< HTTP/1.1 200 OK
< Date: Wed, 27 Jan 2010 19:42:51 GMT
< Server: Apache
< X-Powered-By: ScienceLogic, LLC - EM7 API/Skylar One PowerFlow
< Content-Length: 1087
< Content-Type: application/json
<
[.... REMOVED SEARCHSPEC INFORMATION FROM response, FOR BREVITY]
"total_matched":"1",
```

```
"total_returned":1,

"result_set":[

{

"":{

"URI":"\\api\\discovery_session_active\\1",

"description":"SNMP:1:2"

}

}

]

}

* Connection #0 to host 192.168.10.205 left intact

* Closing connection #0
```

- We receive the HTTP status code and a ScienceLogic status message.
- We found one active discovery session.
- The ID for the active discovery session is "1", with a URI of /api/discovery_session_active/1.

NOTE: Our response is in JSON format. Notice that the URI for the discovery session includes escaped forward slash characters ("V").

Retrieving Information about a Specific Active Discovery-Session

We can request information about a specific, active discovery session using the HTTP GET method with the URI for a specific **discovery_session_active** resource.

To request details about an active discovery session, perform the following:

First, start a discovery session. Enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/discovery_session_active" -H 'content-
type:application/em7-resource-uri' --data-binary "/api/discovery_
session/1"
```

Before the discovery session completes, enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'
"https://192.168.10.205/api/discovery_session_active/1"
```

- The response will be the same as if you requested [all the details about the discovery session](#).
- If the specified discovery_session is not active, the response will include the following:

```
HTTP/1.1 303 See Other
```

```
X-EM7-status-message: The requested discovery_session is not currently
active.
```

```
X-EM7-status-code: FOUND
```

and will also include a redirect to the discovery_session resource for the discovery session.

Viewing Logs for a Discovery Session

When you request information about a discovery session, the returned data includes a **sub-resource**: **logs**. Sub-resources are always associated with their parent resource. Sub-resources have their own URI, appended to that of their parent resource. In our example, **logs** is a sub-resource of a **discovery_session** resource.

If we look at the [response from an HTTP GET of discovery session 1](#), the **logs** sub-resource looks like this:

```
"logs":{
  "URI":"\\api\\discovery_session\\1\\log",
  "description":"Discovery Session Logs"
}
```

Each discovery session has only a single log. Each time the discovery session is executed, the previous log is overwritten with information from the current session.

To view the log for a discovery session, enter the following HTTP GET command at the command prompt:


```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'  
"https://192.168.10.205/api/discovery_session/1/log?limit=100"
```

Note that we appended the URI of the log to the URI of the discovery session, as referenced in the HTTP GET of discovery session 1.

The response looks like this:

```
[...REMOVED CONNECTION INFORMATION AND SOME HEADER INFORMATION FROM  
RESPONSE, FOR BREVITY]
```

```
< HTTP/1.1 200 OK
```

```
< Date: Wed, 27 Jan 2010 20:07:34 GMT
```

```
< Server: Apache
```

```
< X-Powered-By: ScienceLogic, LLC - EM7 API/Skylar One PowerFlow
```

```
< Transfer-Encoding: chunked
```

```
< Content-Type: application/json
```

```
<
```

```
[.... REMOVED SEARCHSPEC INFORMATION FROM response, FOR BREVITY]
```

```
"total_matched":null,
```

```
"total_returned":71,
```

```
"result_set":[
```

```
{
```

```
"log_tstamp":"1264621963",
```

```
"msg_id":"124",
```

```
"msg_txt":"Beginning auto-discovery session"
```

```
},
```

```
{
```

```
"did":"\\device\\113",
```

```
"ip":"192.168.9.71",
```

```
"log_tstamp":"1264621979",  
"msg_id":"500",  
"msg_txt":"Discovered and modeled existing device"  
},  
{  
"did":"\\device\\114",  
"ip":"192.168.9.70",  
"log_tstamp":"1264621979",  
"msg_id":"500",  
"msg_txt":"Discovered and modeled existing device"  
},  
{  
"did":"\\device\\115",  
"ip":"192.168.9.72",  
"log_tstamp":"1264621979",  
"msg_id":"500",  
"msg_txt":"Discovered and modeled existing device"  
},  
{  
"ip":"192.168.9.12",  
"log_tstamp":"1264621994",  
"msg_id":"504",  
"msg_txt":"Discovered, not modeled, pingable device"  
},
```

[...REMOVED REMAINING 66 DEVICE ENTRIES, FOR BREVITY]

```
{  
  
  "log_tstamp":"1264622228",  
  
  "msg_id":"125",  
  
  "msg_txt":"Auto-discovery session completed"  
  
}  
  
]  
  
}  
  
* Connection #0 to host 192.168.10.205 left intact  
  
* Closing connection #0
```

- We receive the HTTP status code and a ScienceLogic status message.
- The log includes an entry for each discovered device, including device IP, device name for SNMP devices, date and time device was discovered, and a description of what was performed on the device.

Stopping a Currently Running Discovery-Session

We can perform an HTTP DELETE method on a **discovery_session_active** resource to stop a currently running discovery session.

Let's first start discovery session 1 again. To do this, enter the following at the command prompt:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'  
"https://192.168.10.205/api/discovery_session_active" -H 'content-  
type:application/em7-resource-uri' --data-binary "/api/discovery_  
session/1"
```

Before the discovery session completes, enter the following at the command prompt to stop the discovery session:

```
curl -v -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword'  
"https://192.168.10.205/api/discovery_session_active/1" -X DELETE
```

- **/api/discovery_session_active/1**. We used the URI of the currently active discovery session.
- **-X DELETE**. We simply appended "-X DELETE" to our HTTP statement to use the DELETE method.

The response looks like this:

```
* About to connect() to 192.168.10.205 port 443 (#0)

* Trying 192.168.10.205... connected

* Connected to 192.168.10.205 (192.168.10.205) port 443 (#0)

* Server auth using Basic with user 'em7admin'

> DELETE /discovery_session_active/1 HTTP/1.1

> Authorization: Basic ZW03YWRTaW46ZW03YWRTaW4=

> User-Agent: curl/7.16.3 (powerpc-apple-darwin9.0) libcurl/7.16.3
OpenSSL/0.9.7l zlib/1.2.3

> Host: 192.168.10.205

> Accept: */*

> X-em7-beautify-response:1

>

< HTTP/1.1 202 Accepted

< Date: Wed, 27 Jan 2010 20:49:59 GMT

< Server: Apache

< X-Powered-By: ScienceLogic,LLC - EM7 API/Skylar One PowerFlow

< X-EM7-status-message: Collector signaled to stop Discovery Session

< X-EM7-status-code: ACCEPTED

< Content-Length: 1

< Content-Type: application/json

<

* Connection #0 to host 192.168.10.205 left intact
```

```
* Closing connection #0
```

- Notice that the ScienceLogic status message indicates that the discovery session will be stopped.
- If the discovery session was not currently running, the response would include the following:

```
HTTP/1.1 400 Bad Request
```

```
X-EM7-status-message: The requested discovery_session is not currently active.
```

```
X-EM7-status-code: BAD_REQ
```

Deleting a Discovery Session

You can use the HTTP DELETE method on a **discovery_session** resource to remove a discovery session from Skylar One. When you remove a discovery session from Skylar One, the entry is removed from the Session Register in the System > Manage > Classic Discovery or System > Manage > Discovery in the classic user interface page, and users can no longer execute this discovery session.

To delete a discovery session from Skylar One, enter the following at the command line:

```
curl -v -k -H 'X-em7-beautify-response:1' -u 'em7admin:examplepassword' "https://192.168.10.205/api/discovery_session/1" -X DELETE
```

- **discovery_session/1**. We used the URI of the discovery session we want to delete.
- **-X DELETE**. We simply appended "-X DELETE" to our HTTP statement, to specify that this is a DELETE method.

Example

3

Searching Component Trees

Overview

The `/device` resource can be filtered using the following fields, which can be used to search a component tree:

- *`component_ancestor_device`*
- *`component_parent_device`*
- *`component_root_device`*
- *`component_unique_id`*

This chapter describes examples of filters that use these fields.

This chapter covers the following topics:

<i>Searching for All the Components in a Tree</i>	<i>183</i>
<i>Searching for the Direct Children of a Device</i>	<i>184</i>
<i>Searching for the Components in a Sub-Tree</i>	<i>185</i>
<i>Searching for a Component by Unique ID</i>	<i>189</i>

Searching for All the Components in a Tree

To search for all the component devices in a tree, i.e., all the component devices under a root device:

- Perform a GET request to the /device resource.
- In the URL of the GET request, include a filter option that matches the component_root_device field to the device ID of the root device.

For example, suppose that you want to get a list of all component devices associated with a NetApp Cluster. Suppose that the NetApp Cluster has a device ID of 1656. To get the list of component devices, you can perform a GET request using the following URL:

```
https://<base URL of the API>/device?limit=100&filter.component_root_device=1656
```

In this example, the result_set in the response looks like this in XML format:

```
<result_set elemtype="list">
```

```
<link URI="/api/device/1673" description="KNT_NetApp_83_C2-01:/vol/vol0"
elemtype="href"/>
```

```
<link URI="/api/device/1697" description="/vol/vserver1_iscsi_
vol2/vserver1_mixed_vol1_lun01" elemtype="href"/>
```

```
<link URI="/api/device/1680" description="vserver1:/vol/vserver1_mix_
vol1" elemtype="href"/>
```

```
<link URI="/api/device/1671" description="data_aggr1" elemtype="href"/>
```

```
<link URI="/api/device/1679" description="vserver1:/vol/vserver1_iscsi_
vol2" elemtype="href"/>
```

```
<link URI="/api/device/1675" description="aggr0_KNT_NetApp_83_C2_02_0"
elemtype="href"/>
```

```
<link URI="/api/device/1687" description="vserver2:/vol/vs2_sm_dest_1"
elemtype="href"/>
```

```
<link URI="/api/device/1678" description="KNT_NetApp_83_C2-02:/vol/vol0"
elemtype="href"/>
```

```
<link URI="/api/device/1660" description="KNT_NetApp_83_C2-01"
elemtype="href"/>
```

```

<link URI="/api/device/1682" description="vserver1:/vol/vs1_sm_dest_1"
elemtype="href"/>

<link URI="/api/device/1663" description="vserver1" elemtype="href"/>

<link URI="/api/device/1681" description="vserver1:/vol/root"
elemtype="href"/>

<link URI="/api/device/1672" description="aggr0" elemtype="href"/>

<link URI="/api/device/1670" description="data_aggr2" elemtype="href"/>

<link URI="/api/device/1662" description="KNT_NetApp_83_C2-02"
elemtype="href"/>

<link URI="/api/device/1677" description="data_aggr4" elemtype="href"/>

<link URI="/api/device/1665" description="vserver2" elemtype="href"/>

<link URI="/api/device/1688" description="vserver2:/vol/root"
elemtype="href"/>

<link URI="/api/device/1676" description="data_aggr3" elemtype="href"/>

<link URI="/api/device/1664" description="vserver3" elemtype="href"/>

<link URI="/api/device/1686" description="vserver3:/vol/root"
elemtype="href"/>

<link URI="/api/device/1661" description="vserver4" elemtype="href"/>

<link URI="/api/device/1674" description="vserver4:/vol/root"
elemtype="href"/>

</result_set>

```

The default response includes the relative API URI and name of each component device. You can add additional options to adjust the response, e.g., the `extended_fetch` option can be used to return all attributes of the component devices in the response.

Searching for the Direct Children of a Device

To search for all the component devices that are direct children of another device, typically another component device:

- Perform a GET request to the /device resource.
- In the URL of the GET request, include a filter option that matches the component_parent_device field to the device ID of the device for which you want to see the children devices.

For example, suppose that you want to get a list of all component devices that are directly associated with an ACI Pod, which includes APIC, Leaf, and Spine devices. Suppose that the ACI Pod has a device ID of 3. To get the list of component devices, you can perform a GET request using the following URL:

```
https://<base URL of the API>/device?limit=100&filter.component_parent_device=3
```

In this example, the result_set in the response looks like this in XML format:

```
<result_set elemtype="list">

  <link URI="/api/device/4" description="Leaf1" elemtype="href"/>

  <link URI="/api/device/5" description="Leaf2" elemtype="href"/>

  <link URI="/api/device/6" description="apic2" elemtype="href"/>

  <link URI="/api/device/7" description="apic3" elemtype="href"/>

  <link URI="/api/device/8" description="apic1" elemtype="href"/>

  <link URI="/api/device/9" description="Spine2" elemtype="href"/>

  <link URI="/api/device/10" description="Spine1" elemtype="href"/>

</result_set>
```

The default response includes the relative API URI and name of each component device. You can add additional options to adjust the response, e.g., the extended_fetch option can be used to return all attributes of the component devices in the response.

Searching for the Components in a Sub-Tree

To search for all the component devices in a sub-tree, i.e., all the component devices under a specific component device:

- Perform a GET request to the /device resource.
- In the URL of the GET request, include a filter option that matches the component_ancestor_device field to the device ID of the root device.

For example, suppose that you want to get a list of all component devices under the US East region component device in an Azure component tree. Suppose that the US East component device has a device ID of 682. To get the list of component devices, you can perform a GET request using the following URL:

```
https://<base URL of the API>/device?limit=100&filter.component_ancestor_device=682
```

In this example, the `result_set` in the response looks like this in XML format:

```
<result_set elemtype="list">
```

```
<link URI="/api/device/693" description="Data & Storage"
elemtype="href"/>
```

```
<link URI="/api/device/694" description="Compute" elemtype="href"/>
```

```
<link URI="/api/device/695" description="Networking" elemtype="href"/>
```

```
<link URI="/api/device/724" description="Storage" elemtype="href"/>
```

```
<link URI="/api/device/725" description="Cloud Services"
elemtype="href"/>
```

```
<link URI="/api/device/726" description="Virtual Machines"
elemtype="href"/>
```

```
<link URI="/api/device/727" description="Virtual Networks"
elemtype="href"/>
```

```
<link URI="/api/device/786" description="portalvhdsr5fxx3bdbnld5"
elemtype="href"/>
```

```
<link URI="/api/device/787" description="temp01tdj" elemtype="href"/>
```

```
<link URI="/api/device/788" description="ywtmpstrgacct"
elemtype="href"/>
```

```
<link URI="/api/device/789" description="wintempeu01" elemtype="href"/>
```

```
<link URI="/api/device/790" description="storagepeu1" elemtype="href"/>
```

```
<link URI="/api/device/791" description="Group Group-10 deletemenettjn"
elemtype="href"/>
```

```
<link URI="/api/device/792" description="VNetPEU1" elemtype="href"/>
```

```
<link URI="/api/device/793" description="virtualnetwork-perm-2"
elemtype="href"/>
```

```
<link URI="/api/device/794" description="Group Api-Default-East-US  
WinTempEU01" elemtype="href"/>
```

```
<link URI="/api/device/825" description="em7-cu3-perm" elemtype="href"/>
```

```
<link URI="/api/device/850" description="em7-cu3-perm" elemtype="href"/>
```

```
<link URI="/api/device/852" description="VmPEA1" elemtype="href"/>
```

```
<link URI="/api/device/853" description="CloudServicePEU2"  
elemtype="href"/>
```

```
<link URI="/api/device/854" description="VmServicePEU1"  
elemtype="href"/>
```

```
<link URI="/api/device/855" description="CloudServicePEU1"  
elemtype="href"/>
```

```
<link URI="/api/device/946"  
description="WADDiagnosticInfrastructureLogsTable" elemtype="href"/>
```

```
<link URI="/api/device/947" description="SchemasTable" elemtype="href"/>
```

```
<link URI="/api/device/948" description="WADMetricSP1MP10DV2S20150720"  
elemtype="href"/>
```

```
<link URI="/api/device/949" description="WADMetricSP1HP10DV2S20150720"  
elemtype="href"/>
```

```
<link URI="/api/device/950" description="WADWindowsEventLogsTable"  
elemtype="href"/>
```

```
<link URI="/api/device/951" description="WADPerformanceCountersTable"  
elemtype="href"/>
```

```
<link URI="/api/device/952" description="vhds" elemtype="href"/>
```

```
<link URI="/api/device/953" description="ywtmptainter"  
elemtype="href"/>
```

```
<link URI="/api/device/957" description="vhds" elemtype="href"/>
```

```
<link URI="/api/device/959" description="vhds" elemtype="href"/>
```

```
<link URI="/api/device/960" description="disks" elemtype="href"/>
```

```
<link URI="/api/device/961" description="vm-images" elemtype="href"/>
```

```
<link URI="/api/device/988" description="WADMetricsPT1MP10DV2S20150630"  
elemtype="href"/>
```

```
<link URI="/api/device/989" description="WADMetricsPT1MP10DV2S20150620"  
elemtype="href"/>
```

```
<link URI="/api/device/990" description="WADMetricsPT1MP10DV2S20150720"  
elemtype="href"/>
```

```
<link URI="/api/device/991" description="WADMetricsPT1MP10DV2S20150710"  
elemtype="href"/>
```

```
<link URI="/api/device/992" description="WADPerformanceCountersTable"  
elemtype="href"/>
```

```
<link URI="/api/device/993" description="LinuxDiskVer2v0"  
elemtype="href"/>
```

```
<link URI="/api/device/994" description="LinuxCpuVer2v0"  
elemtype="href"/>
```

```
<link URI="/api/device/995" description="LinuxsyslogVer2v0"  
elemtype="href"/>
```

```
<link URI="/api/device/996" description="LinuxMemoryVer2v0"  
elemtype="href"/>
```

```
<link URI="/api/device/997"  
description="WADDiagnosticInfrastructureLogsTable" elemtype="href"/>
```

```
<link URI="/api/device/998" description="SchemasTable" elemtype="href"/>
```

```
<link URI="/api/device/999" description="WADMetricsPT1HP10DV2S20150630"  
elemtype="href"/>
```

```
<link URI="/api/device/1000" description="WADMetricsPT1HP10DV2S20150620"  
elemtype="href"/>
```

```
<link URI="/api/device/1001" description="WADMetricsPT1HP10DV2S20150720"  
elemtype="href"/>
```

```

<link URI="/api/device/1778" description="WADMetricsPT1HP10DV2S20151008"
elemtype="href"/>

<link URI="/api/device/1795" description="vmTraffMgrTEU"
elemtype="href"/>

<link URI="/api/device/1796" description="vmTraffMgrTEU"
elemtype="href"/>

<link URI="/api/device/1798" description="TMcloud1" elemtype="href"/>

<link URI="/api/device/1918" description="tempcpuEUSqa"
elemtype="href"/>

<link URI="/api/device/1919" description="tempcpuEUSqa"
elemtype="href"/>

<link URI="/api/device/1920" description="testcpueus01"
elemtype="href"/>

</result_set>

```

The default response includes the relative API URI and name of each component device. You can add additional options to adjust the response, e.g., the `extended_fetch` option can be used to return all attributes of the component devices in the response.

Searching for a Component by Unique ID

To search for a specific component device based on the unique identifier of that component device:

- Perform a GET request to the `/device` resource.
- In the URL of the GET request, include a filter option that matches the `component_unique_id` field to the unique identifier of the component device. The unique identifier format will be different for each type of component device. For example, the unique identifier of an AWS EC2 instance is the instance ID specified by Amazon.
- Typically, you would also use the `extended_fetch` option to return all the attributes of the specified device.

For example, suppose that you want to get all the attributes of an AWS EC2 instance discovered in Skylar One. Suppose that the EC2 instance has an instance ID, which is used by Skylar One as the unique identifier, of `i-c5cf573a`. To get all the attributes of the device, you can perform a GET request using the following URL:

```

https://<base URL of the API>/device?limit=100&filter.component_unique_id=i-c5cf573a&extended_fetch=1

```

In this example, the result_set in the response looks like this in XML format:

```
<result_set elemtype="list">

  <device key="/api/device/74">

    <name>us-east-1a student34: c3.large: i-c5cf573a</name>

    <ip/>

    <hostname elemtype="null"/>

    <snmp_cred_id>/api/credential/snmp/0</snmp_cred_id>

    <snmp_w_cred_id elemtype="null"/>

    <class_type>/api/device_class/451</class_type>

    <collector_group>/api/collector_group/1</collector_group>

    <active>

      <user-disabled>0</user-disabled>

      <unavailable>1</unavailable>

      <maintenance>0</maintenance>

      <system-disabled>0</system-disabled>

      <user-initiated-maintenance>0</user-initiated-maintenance>

    </active>

    <organization>/api/organization/0</organization>

    <auto_update>1</auto_update>

    <event_suppress_mask>00:00:00</event_suppress_mask>

    <auto_clear>1</auto_clear>

    <log_all>1</log_all>

    <daily_port_scan>1</daily_port_scan>

    <critical_ping>0</critical_ping>
```

```

<scan_all_ips>0</scan_all_ips>

<preserve_hostname>1</preserve_hostname>

<disable_asset_update>0</disable_asset_update>

<date_added>1433793323</date_added>

<alert_avail_and_latency_fail>0</alert_avail_and_latency_fail>

<l3_topo elemtype="null"/>

<dashboard elemtype="null"/>

<last_poll elemtype="null"/>

<parent_device elemtype="null"/>

<state>3</state>

<child_devices elemtype="list"/>

<link name="notes" URI="/api/device/74/note/?hide_
filterinfo=1&limit=1000" description="Notes" elemtype="href"/>

<link name="logs" URI="/api/device/74/log/?hide_
filterinfo=1&limit=1000" description="Logs" elemtype="href"/>

<link name="applications" URI="/api/device/74/aligned_app"
description="Aligned Dynamic Applications" elemtype="href"/>

<link name="performance_data" URI="/api/device/74/performance_data"
description="Collected Performance Dynamic App Data" elemtype="href"/>

<link name="config_data" URI="/api/device/74/config_data"
description="Collected Config Dynamic App Data" elemtype="href"/>

<link name="vitals" URI="/api/device/74/vitals"
description="Component-mapped (CPU/MEM/FS) Performance App Data and
Availability/Latency Data" elemtype="href"/>

<link name="interfaces" URI="/api/device/74/interface?limit=1000"
description="Index of interfaces for a device" elemtype="href"/>

```

```
<link name="thresholds" URI="/api/device/74/device_thresholds"
description="Current device threshold values" elemtype="href"/>
```

```
<link name="details" URI="/api/device/74/detail" description="Current
device details" elemtype="href"/>
```

```
<link name="app_credentials" URI="/api/device/74/device_app_
credentials" description="Read-only lookup for aligned credentials and
the device-aligned apps that are using them" elemtype="href"/>
```

```
</device>
```

```
</result_set>
```

Example

4

Simple Provisioning System

Overview

This chapter describes a simple provisioning system written in PHP. The provisioning system is designed to be used by a managed service provider that uses Skylar One to provide monitoring services to its customers.

Using customer information supplied through a simple user interface, the example code makes requests to the API to:

- Create an organization record for the customer.
- Configure SNMP credentials using the supplied community strings.
- Create and run a discovery session.
- Display a list of devices for a specific customer.
- Configure selected devices using device templates.
- Remove a customer from Skylar One by deleting devices, discovery sessions, credentials, and the organization record.

This chapter covers the following topics:

<i>System Design</i>	195
<i>Prerequisites</i>	196
<i>System-Specific Functions</i>	197
<i>Utility Functions (utils.php)</i>	198
<i>User Interface</i>	228
<i>Provisioning a Customer (provision_customer.php)</i>	236

<i>Retrieving and Configuring Devices (configure_devices.php)</i>	244
<i>Removing a Customer (delete_customer.php)</i>	257

System Design

The example provisioning system comprises the following front-end files that display the user interface:

- ***index.php***. Provides a user interface for provisioning a new customer and discovering additional devices for an existing customer.
- ***devices.php***. Provides a user interface for configuring customer devices that have been discovered in Skylar One.
- ***remove.php***. Provides a user interface for removing a customer from Skylar One.

The following back-end files handle the provisioning procedures:

- ***provision_customer.php***. Processes the input values from *index.php* and performs the following provisioning tasks:
 - If an organization record does not currently exist for the customer, creates an organization record for the customer.
 - Configures SNMP credentials for each supplied SNMP community string.
 - Creates a discovery session for the customer using the configured SNMP credentials and the supplied list of IP addresses.
 - Runs the discovery session.

If all of these tasks are successful, *provision_customer.php* redirects to *configure_devices.php*, which will return a list of discovered devices to the *devices.php* page. If a provisioning task is unsuccessful, *provision_customer.php* returns an error message to *index.php*.

- ***configure_devices.php***. The *configure_devices.php* script returns a list of devices and associated device classes for a specified customer. The list of devices can be all devices associated with the customer's organization record, all devices from the last discovery session for that customer, or new devices from the last discovery session for that customer. Additionally, if a user selects the **[Configure Devices]** button in the *devices.php* page, the *configure_devices.php* script applies the device templates selected by the user to the specified devices.
- ***delete_customer.php***. Takes a customer name as input; deletes all devices, credentials, and discovery sessions associated with that customer's organization record; and then deletes the organization record for that customer.

The following diagram shows the control flow between the files when all procedures are successful:

When a back-end procedure is unsuccessful, an error message is returned to the appropriate front-end page.

The six main PHP files use the following additional files:

- **header.php**. Includes the common elements used by all three user interface pages.
- **provisioning.css**. Includes style information for the user interface pages. In this example, minimal style is applied to the user interface pages. You can customize the user interface pages by adding style information to this file.
- **utils.php**. Includes a set of PHP functions that are used by the three back-end files.

Prerequisites

To use the example code described in this chapter to interact with your instance of Skylar One, you must:

- Upgrade your system to version 7.5.5 or later. Some API requests used in the provisioning code are not compatible with older versions of Skylar One.
- Manually create a device template in your instance of Skylar One that will be applied to all devices discovered using the provisioning system.
- Edit `utils.php` to include:

- The IP address of an Administration Portal, Database Server, or All-In-One Appliance in your system.
- An administrator username and password.
- The URI of the device template that will be applied to all devices discovered using the provisioning system.

See the [System-Specific Functions](#) section for a description of the required changes to `utils.php`.

- Copy the example files to a web server. All the example files must be in the same directory on the web server. The web server must:
 - Be able to make HTTP requests to your Administration Portal, Database Server, or All-In-One Appliance.
 - Use a PHP processor module that includes cURL support. The code in this example uses cURL to communicate with an Administration Portal, Database Server, or All-In-One Appliance. For more information about cURL support in PHP, see <http://www.php.net/manual/en/book.curl.php>.
 - Use PHP version 5.2.0 or later. The code in this example uses JSON format for all requests and uses the `json_encode` and `json_decode` functions. For more information about JSON support in PHP, see <http://php.net/manual/en/book.json.php>.
- Manually add a custom attribute to the `/device` resource. The example code uses this custom attribute to track the last device template that was applied to each device. To add the custom attribute, "c-last_dev_tpl", POST the following JSON content to the `/custom_attribute/device` resource index:

```
{  
  
  "name": "last_dev_tpl",
```

```

"label":"last_dev_tpl",

"type":"string",

"index":"none",

"extended":"0"

}

```

For more information about custom attributes, see the [Custom Attributes](#) section.

System-Specific Functions

This example includes two functions in `utils.php` that return information about the instance of Skylar One with which the provisioning code interacts:

- **`get_admin_uri`**. Returns the URL of an Administration Portal, Database Server, or All-In-One Appliance with the username and password of an administrator user embedded in the URL. This value is a required parameter for most functions in *utils.php*.
- **`get_base_template`**. Returns the relative URI of a device template. This device template specifies the basic monitoring parameters for customer devices and is applied to every device discovered using the provisioning system.

To use the example code with your instance of Skylar One, you must edit the **`get_admin_uri`** function to include the IP address of your Administration Portal, Database Server, or All-In-One Appliance, the username for an administrator user, and the password for that administrator user:

```

function get_admin_uri() {

    $is_ip = "10.100.100.15";

    $is_user = "em7admin";

    $is_pass = "<password>";

    $base_uri = "https://" . $is_user . ":" . $is_pass . "@" . $is_ip;

    return $base_uri;
}

```

```
}
```

To use the example code with your instance of Skylar One, you must edit the **get_base_template** function to include the relative URI of a device template in your system:

```
function get_base_template() {  
  
    return "/api/device_template/3";  
  
}
```

Utility Functions (utils.php)

Most tasks performed by the back-end code for this example are performed using a set of generic functions that can be re-used multiple times. If you are developing code that interacts with the ScienceLogic API and are using a different programming language, you might want to start by developing similar generic functions. In this example, the functions are included in the file **utils.php**, which is used by every back-end PHP file. The **utils.php** file includes functions that perform the following procedures:

- *Perform a request to the API using a specified URI, request type, and optional POST content.*
- *Request a list of all entities returned by a specified resource index URI.*
- *Request the URI for an organization record associated with a specified customer name.*
- *Create an entity using a specified set of values.*
- *Delete a list of entities.*
- *Configure a set of SNMP credentials using a specified set of community strings.*
- *Request a list of all devices discovered by a specified discovery session.*
- *Request the URI of a Data Collection Unit in the Collector Group with the most available capacity.*
- *Request a list of entities that are referenced by another list of entities. For example, request a list of device classes associated with a list of devices.*

In addition, **utils.php** includes two functions that return information about the instance of Skylar One with which the provisioning code interacts. For this example, the information returned by these *system-specific functions* is hard-coded.

The following sections describe each function in **utils.php**.

Performing Requests

To perform a request to the ScienceLogic API in PHP, you must:

- Create and configure a cURL session.
- For requests that use the POST method, encode a PHP array as JSON content.

- Execute the cURL request.
- Parse the response and decode the JSON content in to a PHP array.

The **perform_request** function is designed to perform these steps and return the response in a PHP array that has the following structure:

```
(
    ['http_code'] => HTTP status code in the response

    ['headers'] => Array of headers that were included in the response. Each
    key in this array is the name of the header, which points to the value
    for that header.

    ['content'] => Array that contains the decoded JSON body of the
    response.

    ['error'] => If the HTTP code in the response is not healthy (i.e. not
    200, 201, or 202), a human-readable error message that includes all
    error information that was included in the response.
)
```

The **perform_request** function requires the following parameters:

- **\$base**. The URL of an Administration Portal, Database Server, or All-In-One Appliance.
- **\$resource**. The relative URI of the resource to request.

The **perform_request** function has the following optional parameters:

- **\$type**. The type of request to perform. By default, the **perform_request** function performs a GET request (\$type = "GET"). The function accepts the following string values in the \$type parameter:
 - **POST**. The function will POST JSON content to the specified **\$resource**. This method can be used to create or update resources.
 - **APPLY**. The function will POST a resource URI to the specified **\$resource**. This method is used to perform asynchronous operations such as starting discovery sessions and applying device templates to devices. For information about applying a ScienceLogic resource URI to another resource, see the [Asynchronous Operations section](#).
 - **DELETE**. The function will perform an HTTP DELETE request on the specified **\$resource**.
- **\$content**. For **\$type** values that require a POST operation ("POST" or "APPLY"), the content to POST must be passed in this parameter. For a **\$type** value of "POST", **\$content** must be an array, which will be encoded in JSON format. For a **\$type** value of "APPLY", **\$content** must be the relative URI to POST.

The **perform_request** function uses the **\$base** and **\$resource** values to construct the full URI of the resource, then creates a cURL session:

```
function perform_request($base, $resource, $type = "GET", $content = FALSE) {
```

```
    $uri = $base . $resource;
```

```
    $ch = curl_init($uri);
```

For every request, the following cURL options are configured in the cURL session:

- ***CURLOPT_RETURNTRANSFER***. Set to TRUE. By default, the PHP function that executes a request outputs the response to standard out. By specifying this option, the function will return the response as a string.
- ***CURLOPT_HEADER***. Set to TRUE. By specifying this option, the response headers will be included in the output.
- ***CURLOPT_SSL_VERIFYPEER*** and ***CURLOPT_SSL_VERIFYHOST***. Set to FALSE. To enable the use of the example code in a test environment, the verification of the SSL certificate on the API appliance is disabled.

```
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
```

```
    curl_setopt($ch, CURLOPT_HEADER, TRUE);
```

```
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
```

```
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, FALSE);
```

If the ***\$type*** parameter is set to "POST" and content is supplied, the following additional cURL options are set to perform a create/update POST request:

- ***CURLOPT_POST***. Set to TRUE to perform an HTTP POST request.
- ***CURLOPT_POSTFIELDS***. Set to the value of ***\$content*** (in this case, a PHP array) encoded as JSON content.
- ***CURLOPT_HTTPHEADER***. Specifies the appropriate content-type header to include in the request.

```
    if($type == "POST" AND $content) {
```

```
        $json_content = json_encode($content);
```

```
        curl_setopt($ch, CURLOPT_POST, TRUE);
```

```
        curl_setopt($ch, CURLOPT_POSTFIELDS, $json_content);
```

```
        curl_setopt($ch, CURLOPT_HTTPHEADER, array('content-type: application/json'));
```



```
}
```

If the **\$type** parameter is set to "APPLY" and content is supplied, the following additional cURL options are set to perform POST request that applies a resource URI:

- **CURLOPT_POST**. Set to TRUE to perform an HTTP POST request.
- **CURLOPT_POSTFIELDS**. Set to the value of **\$content** (in this case, a the URI of a resource).
- **CURLOPT_HTTPHEADER**. Specifies the appropriate content-type header to include in the request.

```
if($type == "APPLY" AND $content) {  
  
    curl_setopt($ch, CURLOPT_POST, TRUE);  
  
    curl_setopt($ch, CURLOPT_POSTFIELDS, $content);  
  
    curl_setopt($ch, CURLOPT_HTTPHEADER, array('content-type:  
    application/em7-resource-uri'));  
  
}
```

If the **\$type** parameter is set to "DELETE", the **CURLOPT_CUSTOMREQUEST** option is set to perform an HTTP DELETE in the cURL session:

```
if($type == "DELETE") {  
  
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "DELETE");  
  
}
```

If the **\$type** parameter is set to "POST" or "APPLY" and the **\$content** parameter is not supplied, the **perform_request** function returns FALSE without performing a request:

```
elseif(($type == "POST" OR $type == "APPLY") AND !$content) {  
  
    return FALSE;  
  
}
```

The **perform_request** function executes the cURL request and stores the HTTP status code from the response in the output array (***\$response***):

```
$output = curl_exec($ch);
```

```
$response['http_code'] = curl_getinfo($ch, CURLINFO_HTTP_CODE);
```

The response from the API includes the following information that must be included in the output of the function:

- Each response header on a separate line.
- The JSON content in the body of the response on a single line.

To parse this information, an array called ***\$output_array*** is created with each line of the response as an array element. Because the HTTP status code has already been stored, the first line of the response, which contains the HTTP version and status code, is removed from the array:

```
$output_array = explode("\n", $output);
```

```
array_shift($output_array);
```

The function iterates through each line of the response. If a line begins with an opening brace, it is assumed to be the JSON content and is added to the output array (***\$response***):

```
foreach($output_array as $line) {
```

```
    if(strpos($line, "{") < 2 AND strpos($line, "{") !== FALSE) {
```

```
        $response['content'] = json_decode($line, TRUE);
```

```
    }
```

If a line is not content and includes a colon, it is assumed to be a header and is added to the output array (***\$response***):

```
    elseif(strpos($line, ":") !== FALSE) {
```

```
        $header_array = explode(":", $line);
```

```
        $response['headers'][$header_array[0]] = trim($header_array[1]);
```

```
    }
```

```
}
```

To allow other functions to assume that the "content" key always exists in the output array, the "content" key in the output array (*\$response*) is initialized as an empty array if it is not already initialized:

```
if(!array_key_exists('content', $response)) {  
  
    $response['content'] = array();  
  
}
```

In addition to HTTP status codes, every response from the API includes headers that provide additional details about the result of a request:

- ***X-EM7-Implemented-methods***. A comma-delimited list of methods that are supported by the requested resource. This header is intended to provide information on the actions that can be performed on a given resource. For example, if you perform a GET request on the /device resource index, ***X-EM7-Implemented-methods*** will contain "GET,POST", the two methods supported by /device. If you perform a GET request on a specific device (e.g. /device/1), the ***X-EM7-Implemented-methods*** header will contain "GET,POST,PUT,DELETE", because a specific device resource supports all available methods.
- ***X-EM7-Applicable-resources***. A comma-delimited list of base URIs for resources that can be applied to the requested resource. For example, to start a discovery session through the API, you would POST a specific /discovery_session resource to the /discovery_session_active resource index; therefore, if you perform a GET request on the /discovery_session_active resource index, the response will include a ***X-EM7-Applicable-resources*** header of "/discovery_session". For more information on applying resource URIs to other resources, see the [Asynchronous Operations](#) section.
- ***X-EM7-authenticated-user***. The URI of the user account that authenticated the request. If the request included the X-EM7-run-as header, the X-EM7-authenticated-user will return the run-as user.
- ***X-EM7-status-code***. Typically a human-readable version of the HTTP Status Code. For certain errors, ***X-EM7-status-code*** might include additional information about why a request was unsuccessful. For example, if a response has the HTTP Status code "400 Bad Request", the ***X-EM7-status-code*** might be "FAILED_INPUT_VALIDATION".
- ***X-EM7-status-message***. A human-readable description of the result of a request. The ***X-EM7-status-message*** can contain multiple messages delimited by a newline character (`\n`). For example, if a response has the HTTP Status code "302 Found", the ***X-EM7-status-message*** might be "ticket index requires a limit", indicating the request was missing the required **limit** option.
- ***X-EM7-Last-updated***. This header is returned only when requesting device configuration data from the API. Returns the date and time that at least one value in the returned data changed.

If the HTTP status code from the response is not 200, 201, or 202 (i.e. 301 or above), the "error" key in the output array (*\$response*) is set to an appropriate message, which includes the values from the X-EM7-status-message and X-EM7-info-message headers:

```
if($response['http_code'] > 300) {
```

```

$response['error'] = "HTTP status " . $response['http_code'] . "
returned. ";

if(array_key_exists("X-EM7-status-message", $response['headers'])) {

    $response['error'] .= $response['headers']['X-EM7-status-message'] .
    ". ";

}

if(array_key_exists("X-EM7-info-message", $response['headers'])) {

    $response['error'] .= $response['headers']['X-EM7-info-message'] .
    ". ";

}

}

```

Finally, the cURL session is closed and the output array (***\$response***) is returned:

```

curl_close($ch);

return $response;

}

```

Requesting a List of Entities

All resource indexes in the API require the inclusion of the "limit" option in all GET requests; therefore, to obtain a full list of entities from a resource index, you might need to perform multiple requests. For example, if 300 devices are discovered in the system and you use the default limit of "100" when performing a request on the "/device" resource index, you must perform three requests to obtain a list of all devices: one request with an offset of 0, one request with an offset of 100, and one request with an offset of 200.

The **get_all** function is designed to return a list of all available entities for a given resource index URI. The **get_all** function includes a do-while loop that handles cases where multiple requests are required. For example, if the URI is "/device", the **get_all** function returns a list of all devices in the system.

The **get_all** function requires the following parameters:

- **\$base_uri**. The URL of an Administration Portal, Database Server, or All-In-One Appliance.
- **\$uri**. The relative URI of a resource index. The limit and offset parameters are added to the URI by the **get_all** function; the URI must not include limit or offset parameters. The logic in the **get_all** function requires that responses from the API include the **total_matched** value; therefore, the passed URI must not include the **hide_filterinfo** parameter.

The **get_all** function returns:

- On success, an array of entities. The structure of the array of entities is identical to the structure returned in the **result_set** section of the response from the specific resource URI. The array of entities can be empty if the request to the resource URI was successful, but no results were returned.
- On failure, an error message.

Any function that calls the **get_all** function can check success/failure by determining if the returned value is an array or a string.

Before executing the do-while loop in which requests to the resource URI are performed, the array of entities is initialized, initial offset value is set to 0, and the limit and offset values are added to the URI:

```
function get_all($base_uri, $uri) {  
  
    $offset = 0;  
  
    $request_uri = $uri . "&limit=100&offset=";  
  
    $entities = array();
```

The the **\$request_uri** variable does not include a value for the offset option. For each iteration of the do-while loop, the current offset is appended to the end of **\$request_uri**.

The do-while loop performs a GET request for the URI with the current offset. If the request was successful (the HTTP status code is 200), the **result_set** from the request is added to the list of entities:

```
do {  
  
    $response = perform_request($base_uri, $request_uri . $offset, "GET");  
  
    if($response['http_code'] == 200 AND array_key_exists("result_set",  
    $response['content']) AND count($response['content']['result_set']) >  
    0) {  
  
        $entities = array_merge($entities, $response['content']['result_  
        set']);
```

```
}
```

If the request is unsuccessful, the ***\$message*** variable is initialized with an error message:

```
elseif($response['http_code'] != 200) {  
  
    $message = "An error occurred while requesting entities. ";  
  
    if(array_key_exists("error", $response)) {  
  
        $message .= $response['error'];  
  
    }  
  
}
```

Because the limit parameter is set to 100 in the URI, the offset value is incremented by 100 on each iteration. The do-while loop iterates if the previous request was successful and more entities are available. The "total_matched" value from the previous response indicates the total number of entities that can be returned by this specific URI; more entities are available if the current offset value is lower than "total_matched":

```
$offset = $offset + 100;  
  
} while(!isset($message) AND ($offset < $response['content']['total_  
matched']));
```

If an error message was generated by any request performed by the **get_all** function, the returned value is the error message generated by the failed request. If no error messages were generated, the array of entities is returned:

```
if(isset($message)) {  
  
    return $message;  
  
}  
  
else {  
  
    return $entities;  
  
}
```

```
}  
  
}
```

Organization Lookup

The input forms used in this example include a field for customer name. The **lookup_organization** function is designed to return the URI for a customer's organization record using the name of a customer.

The **lookup_organization** function requires the following parameters:

- ***\$base_uri***. The URL of an Administration Portal, Database Server, or All-In-One Appliance.
- ***\$customer***. The customer name.

The **lookup_organization** function returns:

- On success, the URI of the organization record for the specified customer.
- On failure, boolean FALSE.

The **lookup_organization** function constructs a request to the /organization resource index using the customer name as a search parameter. The customer name is URL encoded to handle names that include spaces:

```
function lookup_organization($base_uri, $customer) {  
  
    $resource = "/api/organization?limit=1&hide_  
filterinfo=1&filter.company=" . rawurlencode($customer);  
  
    $response = perform_request($base_uri, $resource, "GET");
```

If the request was successful (the HTTP status code is 200) and at least one organization is returned, the URI of the first organization in the response is returned. Because the request specified that the customer name must be matched exactly and because all organization names in an instance of Skylar One must be unique, the assumption is made that the response will not include more than one organization:

```
    if($response['http_code'] == 200 AND count($response['content']) > 0 AND  
    array_key_exists("URI", $response['content'][0])) {  
  
        return $response['content'][0]['URI'];  
  
    }  
  
    else {  
  
        return FALSE;
```

```
}  
  
}
```

Creating Entities

The **create_entity** function is designed to create an entity using the resource index URI for that entity and an array of field/value pairs for the entity.

The **create_entity** function requires the following parameters:

- ***\$base_uri***. The URL of an Administration Portal, Database Server, or All-In-One Appliance.
- ***\$entity_uri***. The resource index URI for the entity to be created. For example, to create an organization, supply `/api/organization` in the ***\$entity_uri*** parameter.
- ***\$entity_array***. A PHP array that contains field/value pairs of the attributes for the entity. This PHP array will be converted to JSON format and POSTed to the specified URI.

The **create_entity** function returns an array:

- The first array value (array index 0) is a boolean that indicates whether the entity was created successfully.
- The second array value (array index 1) is a string. On success, the string is the URI of the created entity. On failure, the string is an error message.

The **create_entity** function uses the **perform_request** function to create the entity. The **perform_request** function handles the conversion of the PHP array to JSON format and the options required to perform a POST request:

```
function create_entity($base_uri, $entity_uri, $entity_array) {  
  
    $response = perform_request($base_uri, $entity_uri, "POST", $entity_  
    array);
```

If the request was successful (the HTTP status code is 201), the function returns TRUE at array index 0 and the contents of the "Location" header at array index 1, which contains the relative URI of the created element:

```
    if($response['http_code'] == 201) {  
  
        return array(TRUE, $response['headers']['Location']);  
  
    }
```


If the request was unsuccessful, the function returns FALSE at array index 0 and the error message at array index 1:

```
else {  
  
    $error_message = "Could not create " . substr($entity_uri, 1) . ". ";  
  
    if(array_key_exists("error", $response)) {  
  
        $error_message .= $response['error'];  
  
    }  
  
    return array(FALSE, $error_message);  
  
}  
  
}
```

Deleting Entities

The **multi_delete** function is designed to delete multiple entities.

The **multi_delete** function requires the following parameters:

- **\$base_uri**. The URL of an Administration Portal, Database Server, or All-In-One Appliance.
- **\$entities**. An array that contains the entities to be deleted. The **\$entities** array must be multi-dimensional; each element in the **\$entities** array must be an array that includes "URI" as a key. The function uses the value of "URI" as the relative URI in a delete request. The structure of the **\$entities** array is the same as an array returned by the [get_all](#) function.

The **multi_delete** function returns NULL on success or an error message on failure.

The **multi_delete** function initializes **\$bad_entities** as an array. The **\$bad_entities** array is used to track entities that could not be deleted:

```
function multi_delete($base_uri, $entities) {  
  
    $bad_entities = array();
```

If the input is valid (**\$entities** is an array), the **multi_delete** function iterates through each element in the array. For each element, if the element is an array that contains the key "URI", the function performs a delete request using the value that corresponds to the key "URI". If the element was not an array, did not contain the key "URI", or the delete request fails, the element is added to the **\$bad_entities** array:

```
if(is_array($entities)) {
```

```

foreach($entities as $entity) {

    if(is_array($entity) AND array_key_exists('URI', $entity)) {

        $response = perform_request($base_uri, $entity['URI'], "DELETE");

        if($response['http_code'] >= 400) {

            $bad_entities[] = $entity;

        }

    }

    else {

        $bad_entities[] = $entity;

    }

}

```

If all elements in the ***\$entities*** array were deleted, the **multi_delete** function returns NULL, indicating success:

```

if(count($bad_entities) == 0) {

    return NULL;

}

```

If one or more elements in the ***\$entities*** array could not be deleted, an error message is constructed by concatenating the contents of each element in ***\$entities*** that could not be deleted. Instead of determining the data type of each element, the **print_r** function is used to output the human-readable string for the element:

```

else {

    $error_message = "Could not delete: ";

    foreach($bad_entities as $entity) {

        $error_message .= print_r($entity, TRUE) . ". ";

    }

}

```

```

        return $error_message;
    }
}

else {

    return "Must pass an array of entities";

}

}

```

Configuring SNMP Credentials

The **configure_credentials** function is designed to return an array of SNMP v2 credentials for a specific organization using a list of community strings. The **configure_credentials** function creates new credentials if a credential with the same community string does not already exist for the organization.

The **configure_credentials** function requires the following parameters:

- ***\$base_uri***. The URL of an Administration Portal, Database Server, or All-In-One Appliance.
- ***\$customer***. The name of the customer organization for which the credentials will be used.
- ***\$community_strings***. A comma-delimited list of community strings. The **configure_credentials** function ensures that a credential associated with the ***\$customer*** organization exists for each community string in the list.

The **configure_credentials** function returns an array of credential URIs on success or an error message on failure.

The **configure_credentials** function uses the **array_walk** PHP function when the list of community strings is parsed. The **array_walk** function takes the name of a function as a parameter and applies that function to each value in the array. In our example code, the **array_walk** applies the **trim_value** function to each value in the array. The **trim_value** function is included in the **utils.php** file and removes leading and trailing whitespace from each value passed in the parameter:

```

function trim_value(&$value) {

    $value = trim($value);

}

```

The ***\$community_strings*** parameter is split into an array of community strings. If a user enters spaces in the comma-delimited list, the **trim_value** function removes leading and trailing whitespace from each element in the array:

```
function configure_credentials($base_uri, $customer, $community_strings) {  
  
    $community_array = explode(",", $community_strings);  
  
    array_walk($community_array, 'trim_value');
```

All credentials created by the **configure_credentials** function are named "customer: community string". The **configure_credentials** function performs a request for all credentials associated with the specified customer by searching for credential names that include the string ***\$customer***.

```
$resource = "/api/credential/snmp?limit=1000&snmp_version=2&hide_  
filterinfo=1&filter.cred_name.contains=" . rawurlencode($customer);  
  
$existing_credentials = perform_request($base_uri, $resource, "GET");
```

If the request for existing credentials is successful, the response is processed using the following arrays:

- ***\$existing_credentials***. The response to the request for all credentials currently associated with the organization specified in ***\$customer***.
- ***\$community_array***. The array of community strings passed in the ***\$community_strings*** parameter.
- ***\$existing_communities***. Initialized to an empty array. As the function iterates through ***\$existing_credentials***, the community string for each existing credential that matches a community string that was passed in the ***\$community_strings*** parameter is added to this array.
- ***\$credentials***. Initialized to an empty array. If a community string for an existing credential matches a community string that was passed in the ***\$community_strings*** parameter, the URI for that credential is added to this array.

```
if($existing_credentials['http_code'] == 200) {
```

```
    $credentials = array();
```

```
    $existing_communities = array();
```

The **configure_credentials** function iterates through the existing credentials for the organization in the ***\$existing_credentials*** array. The community string is parsed from the name of the existing credential based on the standard naming scheme. If the community string matches a value in ***\$community_array***, the community string is added to the ***\$existing_communities*** array and the URI is added to the ***\$credentials*** array:

```
foreach($existing_credentials['content'] as $key => $credential) {
```

```

$existing_community = substr($credential['description'], strlen
($customer) + 2);

$matched_community = array_search($existing_community, $community_
array);

if($matched_community !== FALSE) {

    $credentials[] = $credential['URI'];

    $existing_communities[] = $community_array[$matched_community];

}

}

```

The **configure_credentials** function must now create a credential for any community string that appears in ***\$community_array*** that does not appear in ***\$existing_communities***. The variable ***\$error_message*** is initialized as an empty string; all error messages generated while credentials are added are appended to this string. The variable ***\$diff*** is initialized as an array of community strings that appear in ***\$community_array*** that do not appear in ***\$existing_communities***:

```

$error_message = "";

$diff = array_diff($community_array, $existing_communities);

```

If ***\$diff*** is empty, i.e. no additional credentials need to be created, processing is complete. If new credentials need to be created, the variable ***\$organization*** is initialized to the URI of the organization record associated with ***\$customer***:

```

if(count($diff) > 0) {

    $organization = lookup_organization($base_uri, $customer);

```

If the organization URI is returned by the **lookup_organization** function, the **configure_credentials** function iterates through the community strings in ***\$diff***. For each community string, the credential name is constructed using the customer name and the community string:

```

if($organization !== FALSE) {

    foreach($diff as $community) {

```

```
$cred_name = $customer . ": " . trim($community);
```

The variable ***\$cred_post_array*** is initialized to an array that represents the content that will be used to create the credential. When the credential is created, the **create_entity** function encodes this array in JSON format. The array includes the following field/value pairs that are applicable to /credential/snmp resources:

```
(  
  
  ['cred_name'] => The name of the credential.  
  
  ['cred_host'] => The hostname associated with the credential. Always set  
  to an empty string.  
  
  ['cred_port'] => The port associated with the credential. Always set to  
  the standard SNMP port, 161.  
  
  ['cred_timeout'] => The timeout for the credential. Always set to a  
  default timeout of 1500ms.  
  
  ['all_orgs'] => This setting specifies whether the credential is visible  
  to all organizations (1) or is restricted to specific organizations (0).  
  All credentials created by the provisioning system are aligned only with  
  the specific organization for which they are created, so this value is  
  always set to 0.  
  
  ['snmp_version'] => The SNMP version. For simplicity, this example  
  creates only SNMP v2 credentials.  
  
  ['snmp_ro_community'] =>The SNMP community string.  
  
  ['aligned_organizations'] => A list of organizations to which the  
  credential is visible. A list element in JSON is represented as an array  
  in the equivalent PHP structure.  
  
)
```

The ***\$cred_post_array*** variable is passed to the **create_entity** function with the URI of an Administration Portal, Database Server, or All-In-One Appliance and the relative URI that is used to create SNMP credentials (/api/credential/snmp):

```

$ccred_post_array = array('cred_name' => $cred_name, 'cred_host'
=> "", 'cred_port' => 161, 'cred_timeout' => 1500, 'all_orgs' =>
0, 'snmp_version' => 2, 'snmp_ro_community' => trim($community),
'aligned_organizations' => array($organization));

$cred_response = create_entity($base_uri,
"/api/credential/snmp", $ccred_post_array);

```

The **create_entity** function returns an array of two values. Index 0 in the returned array is a boolean that indicates whether the entity was created successfully. Index 1 in the returned array is the URI of the created entity on success or an error message on failure. If the credential was created successfully, the URI of the new credential is added to the **\$credentials** array. If the credential was not created, the error message from the **create_entity** function is appended to **\$error_message**:

```

if($cred_response[0]) {

    $credentials[] = $cred_response[1];

}

else {

    $error_message .= $cred_response[1];

}

}

}

```

If no organization URI was returned by the **lookup_organization** function, an error message is appended to **\$error_message**:

```

else {

    $error_message .= "Could not find organization record for
customer: " . $customer . ". ";

}

}

}

```

If the request for existing credentials is not successful, an the ***\$error_message*** variable is set to an error message that includes the error message constructed by the **perform_request** function, if available:

```
else {  
  
    $error_message = "Could not get list of existing credentials. ";  
  
    if(array_key_exists("error", $existing_credentials)) {  
  
        $error_message .= $existing_credentials['error'] . ". ";  
  
    }  
  
}
```

If an error message has been generated by the **create_credentials** function, that error message is returned. Otherwise, the array of credential URIs is returned:

```
if(strlen($error_message) == 0) {  
  
    return $credentials;  
  
}  
  
else {  
  
    return $error_message;  
  
}  
  
}
```

Requesting Discovery Session Logs

The **get_discovery_result** function is designed to return an array that contains information about a specified discovery session. The returned array has the following structure:

```
(  
  
    ['status'] => An integer that specifies the result of the get_discovery_  
    result function:
```


0 = The specified discovery session has completed and `get_discovery_result` was able to return a list of devices discovered by the discovery session.

1 = The specified discovery session is currently running and `get_discovery_result` was able to return a list of devices discovered by the discovery session.

2 = The specified discovery session has never been run.

3 = An error occurred in a request made by the `get_discovery_result` function.

`['devices']` => If the returned status is 0 or 1, is set to an array of device arrays. Each device array includes "ip", "name", "uri", and "new" keys. The "new" key is a boolean that is set to TRUE if the device was discovered as a new device or FALSE if the device was discovered as an existing device.

`['error']` => If the returned status is 3, is set to an error message.

)

The `get_discovery_result` function requires the following parameters:

- **`$base_uri`**. The URL of an Administration Portal, Database Server, or All-In-One Appliance.
- **`$session_uri`**. The URI for a discovery session resource.

The `get_discovery_result` has the following optional parameter:

- **`$new_only`**. If TRUE is passed in this parameter, the list of devices returned by the function will include only newly discovered devices from the discovery session. By default, the function returns all discovered devices, both new and existing, from the discovery session.

```
function get_discovery_result($base_uri, $session_uri, $new_only = FALSE)
{
```

The function includes a do-while loop in which all log messages for a discovery session are requested. Like the `get_all` function, a limit of 100 is specified in the logs URI and the offset is increased on each iteration of the do-while loop. The variable **`$discovery_logs`** is initialized as an array, to which the log

messages in the responses will be added. The variable ***\$not_started*** is initialized as FALSE. If the logic within the do-while loop determines that the discovery session is not running, this variable is set to TRUE:

```
$log_uri = $session_uri . "/log?extended_fetch=1&limit=100&offset=";
```

```
$offset=0;
```

```
$discovery_logs = array();
```

```
$not_started = FALSE;
```

```
do {
```

```
    $response = perform_request($base_uri, $log_uri . $offset, "GET");
```

If the request for logs in this iteration of the do-while loop successfully returns logs, the returned logs are added to the ***\$discovery_logs*** array:

```
if($response['http_code'] == 200 AND array_key_exists("result_set",  
$response['content']) AND count($response['content']['result_set'] >  
0) {
```

```
    $discovery_logs = array_merge($discovery_logs, $response['content']  
    ['result_set']);
```

```
}
```

If the request for logs is successful but does not return any logs, the function must determine whether the discovery session was never started or if the discovery session is running but has not yet generated any logs. To do this, the URI of the discovery session is manipulated to determine the equivalent `/api/discovery_session_active` URI:

```
elseif($response['http_code'] == 200 AND array_key_exists("total_  
matched", $response['content']) AND $response['content']['total_  
matched'] == 0) {
```

```
    $uri_array = explode("/", $session_uri);
```

```
    $uri_array[2] = "discovery_session_active";
```

```
    $active_uri = implode("/", $uri_array);
```

The function performs a GET request on the `/discovery_session_active` URI for the specified discovery session. If the response includes an HTTP status code of 200, the discovery session is currently running. The output array (***\$result***) is initialized with a status of 1 (running) with an empty array of devices:

```
$active_check = perform_request($base_uri, $active_uri, "GET");
```

```
if($active_check['http_code'] == 200) {
```

```
    $result = array("status" => 1, "devices" => array());
```

```
}
```

If the response includes an HTTP status code of 303 (See Other), the discovery session exists but is not currently running. The output array (***\$result***) is initialized with a status of 2 (never run) and an appropriate error message:

```
elseif($active_check['http_code'] == 303) {
```

```
    $result = array("status" => 2, "error" => "Discovery Session has  
    never run.");
```

```
}
```

If the response includes an HTTP status code other than 200 or 303, an error occurred with the request. The output array (***\$result***) is initialized with a status of 3 (error) and an appropriate error message:

```
else {
```

```
    $result = array("status" => 3, "error" => "Could not determine  
    status of discovery session. ");
```

```
    if(array_key_exists("error", $active_check)) {
```

```
        $result['error'] .= $active_check['error'];
```

```
    }
```

```
}
```

```
}
```

If the request for discovery session logs fails (HTTP status code is not 200), the output array (***\$result***) is initialized with a status of 3 (error) and an appropriate error message:

```
else {  
  
    $result = array("status" => 3, "error" => "Could not get discovery  
    session logs ");  
  
    if(array_key_exists("error", $response)) {  
  
        $result['error'] .= $response['error'];  
  
    }  
  
}
```

The offset is increased for the next iteration for the do-while loop. The loop continues if the output array (***\$result***) has not been initialized, i.e. the request for logs was successful and returned one or more logs, and if more logs are available. The "total_matched" value from the previous response indicates the total number of logs that can be returned; more logs are available if the current offset value is lower than "total_matched":

```
$offset = $offset + 100;  
  
} while(!isset($result) AND array_key_exists("total_matched", $response  
['content']) AND ($offset < $response['content']['total_matched']));
```

If the output array (***\$result***) has not been initialized, all requests performed in the do-while loop were successful and one or more logs were returned. In this case, the status returned by the **get_discovery_result** function will be either 0 (logs were successfully returned and the discovery session is complete) or 1 (logs were successfully returned and the discovery session is still running). The function iterates through the array of returned log messages:

```
if(!isset($result)) {  
  
    $result = array("devices" => array());  
  
    foreach($discovery_logs as $log) {
```

Each discovery session log includes a "msg_id" field, which specifies the type of message in the log entry. To return a list of devices and to determine the state of the discovery session, the **get_discovery_results** function uses only log messages that have one of the following msg_id values:

- **125**. Associated with the log message that indicates the discovery session is complete.
- **500**. Associated with the log message that is generated when an existing device is found by the discovery session.
- **501**. Associated with the log message that is generated when a new device is found by the discovery session.

The "msg_id" field is used in a switch statement, which includes cases for the three values:

```
switch($log['msg_id']) {
```

If the log message indicates the discovery session is complete, the status key in the output array is set to 0:

```
case 125:
```

```
$result['status'] = 0;
```

```
break;
```

If the log message indicates an existing device is found and the **\$new_only** parameter is set to FALSE, the device is added to the device array:

```
case 500:
```

```
if(!$new_only) {
```

```
$result['devices'][] = array("ip" => $log['ip'], "name" =>  
$log['name'], "uri" => $log['device'], "new" => FALSE);
```

```
}
```

```
break;
```

New devices are always added to the device array:

```
case 501:
```

```
$result['devices'][] = array("ip" => $log['ip'], "name" => $log  
['name'], "uri" => $log['device'], "new" => TRUE);
```

```

        break;
    }
}

```

If the status key in the output array has not been set after all log messages have been evaluated, the discovery session is still running:

```

    if(!array_key_exists("status", $result)) {

        $result['status'] = 1;

    }

}

return $result;

}

```

Requesting an Available Data Collection Unit

To create a discovery session using the API, you must specify the URI of an /appliance resource. The supplied /appliance resource must be an All-In-One Appliance or a Data Collector. The **get_collector_id** function is designed to return the URI of an appliance for discovery.

The **get_collector_id** function requires the following parameter:

- ***\$base_uri***. The URL of an Administration Portal, Database Server, or All-In-One Appliance.

The **get_collector_id** function returns an array:

- The first array value (array index 0) is a boolean that indicates whether an appropriate appliance resource was found.
- The second array value (array index 1) is a string. On success, the string is the URI of an appropriate appliance. On failure, the string is an error message.

For systems that include All-In-One Appliances, the function returns the URI of the currently active All-In-One Appliance. For distributed systems, the function returns the URI of a Data Collector in a collector group.

An initial request is made on the /appliance resource index. The request includes filter criteria that specifies that only All-In-One Appliances (type = "ao") that are currently active (ha_status = 1) should be returned:

```
function get_collector_id($base_uri, $num_devices) {
```

```
    $resource = "/api/appliance?limit=100&filter.type=ao&filter.ha_
    status=1&hide_filterinfo=1";
```

```
    $response = perform_request($base_uri, $resource, "GET");
```

If the response includes at least one appliance, the URI of that appliance is returned:

```
    if($response['http_code'] == 200 AND count($response['content']) > 0) {
        return array(TRUE, $response['content'][0]['URI']);
    }
```

If the initial request fails, the function returns an error message:

```
    elseif($response['http_code'] != 200) {
        $error_message = "Request for list of appliances failed. ";
        if(array_key_exists("error", $response)) {
            $error_message .= $response['error'];
        }
        return array(FALSE, $error_message);
    }
```

If the initial request is successful, but does not return any appliances, a request is made for all collector groups in the system using the extended fetch option:

```
else {  
  
    $resource = "/api/collector_group?limit=100&hide_  
    filterinfo=1&extended_fetch=1";  
  
    $response = perform_request($base_uri, $resource, "GET");
```

If the request for collector groups is successful and at least one collector group is returned, the function iterates through the array of returned collector groups. For each collector group, the function checks the `data_collectors` field. If a collector group includes at least one Data Collector, the URI of the first Data Collector in that collector group is returned:

```
if($response['http_code'] == 200 AND count($response['content']) > 0)  
{  
  
    foreach($response['content'] as $cug_id => $cug) {  
  
        if(array_key_exists("data_collectors", $cug) AND count($cug['data_  
        collectors']) > 0) {  
  
            return array(TRUE, $cug['data_collectors'][0]);  
  
        }  
  
    }  
  
}  
  
}
```

If the request for collector groups is not successful (the HTTP Status Code in the response is not 200), an appropriate error message is returned. If an error message was returned by the `perform_request` function, it is included in the error message:

```
elseif($response['http_code'] != 200) {  
  
    $error_message = "Request for list of collector groups failed. ";  
  
    if(array_key_exists("error", $response)) {  
  
        $error_message .= $response['error'];  
  
    }  
  
}
```



```

        return array(FALSE, $error_message);
    }

```

If the request for collector groups is successful but does not return any collector groups, an appropriate error message is returned:

```

    else {

        return array(FALSE, "No collector groups configured on system.");
    }

}

}

```

Requesting a List of Referenced Entities

API resources that represent a specific entity can include references to other entities. These references are displayed as the relative URI of that other entity. For example, if you perform a GET request on `/api/device/1`, the response will include a `"class_type"` field that contains the URI of the device class associated with the device. The **get_join_resources** function is designed to return an array of entities referenced in a particular field in a passed array of entities.

The **get_join_resources** function requires the following parameters:

- ***\$base_uri***. The URL of an Administration Portal, Database Server, or All-In-One Appliance.
- ***\$entity_list***. An array that contains the entities that include a field that references another entity. For example, if you want to retrieve an array of device classes that are associated with a set of devices, you would pass the array of devices in the ***\$entity_list*** parameter.
- ***\$left_join_field***. The name of the field associated with the entities in the ***\$entity_list*** that reference the other entity. For example, all devices include a `"class_type"` field that specifies the URI of the device class associated with that device; therefore, if you want to retrieve an array of device classes that are associated with a set of devices, you would pass `"class_type"` in the ***\$left_join_field*** parameter.

- ***\$right_join_field***. For efficiency, the **get_join_resources** function performs a single request for all referenced entities instead of performing a request on each referenced URI. To request all referenced entities, the **get_join_resources** function performs a request to the appropriate resource index with the **extended_fetch** option enabled. To limit the request to only entities that are referenced by the entities in the ***\$entity_list***, the **get_join_resources** function concatenates the ID values of each referenced entity and passes them as a search value using the "in" function. To do this, the **get_join_resources** function must specify the field in the referenced entity that contains the ID value. You must pass the name of that field in the ***\$right_join_field*** parameter. For example, if you want to retrieve an array of device classes that are associated with devices and the **get_join_resources** function determines that the devices in ***\$entity_list*** are associated with device classes 2, 5, and 9, the **get_join_resources** function must pass the following URI to the **get_all** function to get the list of device classes:

```
/api/device_class?extended_fetch=1&filter.class_type.in=2,5,9
```

The field used in the filter clause must be passed in the ***\$right_join_field*** parameter. In this case, ***\$right_join_field*** is "class_type".

The **get_join_resources** function returns an array of entities on success or an error message on failure. If no referenced entities are found, the returned array is empty.

The **get_join_resources** function initializes ***\$in*** as an array. The ***\$in*** array is used to track the ID values of the entities that must be requested:

```
function get_join_resources($base_uri, $entity_list, $left_join_field,
    $right_join_field) {

    $in = array();
```

The **get_join_resources** function iterates through the array of entities passed in the ***\$entity_list*** parameter. For each entity in the array, the function looks up the URI for the referenced entity using the field name passed in the ***\$left_join_field*** parameter. Two variables are populated by the URI of the referenced entity:

- ***\$join_uri***. Initialized to an array that contains each section of the URI (delimited by the "/" character) as an element. The specific resource ID is removed from the end of the array using the **array_pop** function; therefore, when the foreach loop completes, ***\$join_uri*** is an array that contains each section of the URI for the resource index of the referenced entity.

For example, if the URI of the referenced entity is `/credential/snmp/1`, the ***\$join_uri*** array looks like this when the foreach loop completes:

```
(  
    [0] => "credential"  
    [1] => "snmp"  
)
```

- ***\$in***. An element in this array is set to the ID value of the referenced entity.

```
foreach($entity_list as $entity) {  
    if(array_key_exists($left_join_field, $entity)) {  
        $join_uri = explode("/", $entity[$left_join_field]);  
        $in[] = array_pop($join_uri);  
    }  
}
```

If at least one entity ID exists in the ***\$in*** array, the **`get_join_resources`** function constructs a URI for the resource index of the referenced entities using:

- The ***\$join_uri*** array, which contains the base resource index URI.
- The value in ***\$right_join_field***, which is the field in the referenced entity that contains the ID values for that entity.
- The ***\$in*** array, which contains the ID values of each referenced entity from the list of entities that was passed in the ***\$entity_list*** array.

The URI is used to request a list of referenced entities:

```
if(count($in) > 0) {  
    $uri = implode("/", $join_uri) . "?extended_fetch=1&filter." . $right_  
    join_field . ".in=" . implode(",", $in);  
    $join_list = get_all($base_uri, $uri);  
}
```

The **get_all** function returns an array or entities on success and an error message on failure. Because the **get_all** function returns the same values as the **get_join_resources** function, the result of the **get_all** function can be returned without additional processing:

```
        return $join_list;
    }
}
```

If no entity IDs exist in the **\$in** array, the function returns an empty array without performing a request:

```
    else {
        return array();
    }
}
```

User Interface

The example provisioning system comprises the following front-end files to display the user interface:

- **index.php**. Provides a user interface for provisioning a new customer and discovering additional devices for an existing customer.
- **devices.php**. Provides a user interface for configuring customer devices that have been discovered in Skylar One.
- **remove.php**. Provides a user interface for removing a customer from Skylar One.

The user interface files use the following additional files:

- **header.php**. Includes the common elements used by all three user interface pages.
- **provisioning.css**. Includes style information for the user interface pages. In this example, minimal style is applied to the user interface pages. You can customize the user interface pages by adding style information to this file.

The following sections describe each of these five files.

header.php

The **header.php** file is required in all user interface PHP files. The **header.php** file outputs links to each user interface page, includes **utils.php**, and outputs messages from the back-end PHP files:

```
<p><a href="index.php">Run Discovery</a> | <a href="devices.php">Configure
Devices</a> | <a href="remove.php">Remove Customer</a></p>
```

```

<?php

require_once 'utils.php';

session_start();


if(isset($_SESSION['message'])) {

    echo "<p>" . $_SESSION['message'] . "</p>";

    unset($_SESSION['message']);

}

else {

    echo "<p>&nbsp;</p>";

}

?>

```

index.php

The ***index.php*** file provides a user interface for provisioning a new customer and discovering additional devices for an existing customer:

```

<html>

<head>

    <title>Provision Customer</title>

    <link href="provisioning.css" rel="stylesheet" type="text/css">

</head>

<body>

    <?php

        require_once 'header.php';

```

```

?>

<form action="provision_customer.php" method="post">

    Customer Name:<br />

    <input type="text" name="customer" /><br />

    Device IP List (Comma-separated):<br />

    <input type="text" name="ip_addresses" /><br />

    SNMP v2 Community Strings:<br />

    <input type="text" name="community_strings" /><br />

    <input type="checkbox" name="non_snmp" value="yes" /> Discover Non-
    SNMP Devices<br />

    <input type="submit" value="Submit" />

</form>

</body>

</html>

```

When you enter customer information and select the **[Submit]** button, the [provision_customer.php script](#) performs the required tasks for provisioning that customer.

devices.php

The *devices.php* file provides a user interface for configuring customer devices in Skylar One:

When you enter a customer name and select the **[Show Devices]** button, the [configure_devices.php script](#) returns a list of devices that are then displayed in the *devices.php* page. For each device in the list of devices, the *devices.php* page displays a drop-down list of all device templates in the **Service Level** column. The last device template that was applied to the device is selected by default:

```

<html>

<head>

```

```

<title>Configure Devices</title>

<link href="provisioning.css" rel="stylesheet" type="text/css">

</head>

<body>

<?php

    require_once 'header.php';

```

The **configure_devices.php** back-end script returns session variables that contain the values that were supplied in the **Customer Name** field and radio buttons. These session variables are stored in a local variable then unset. The local variables are used to populate the form fields:

```

if(isset($_SESSION['customer'])) {

    $customer = $_SESSION['customer'];

    unset($_SESSION['customer']);

}

else {

    $customer = "";

}

if(isset($_SESSION['dev_type'])) {

    $dev_type = $_SESSION['dev_type'];

    unset($_SESSION['dev_type']);

}

else {

    $dev_type = "";

}

?>

```

```

<p>Devices To Configure:</p>

<form action="configure_devices.php" method="post">

Customer Name:

<input type="text" name="customer" value="<?php echo $customer; ?>"
/><br />

<input type="radio" name="dev_type" value="new_disc" <?php if($dev_
type == "new_disc") echo checked; ?> /> New Devices from last
Discovery<br />

<input type="radio" name="dev_type" value="all_disc" <?php if($dev_
type == "all_disc") echo checked; ?> /> All Devices from last
Discovery<br />

<input type="radio" name="dev_type" value="all_org" <?php if($dev_type
== "all_org") echo checked; ?> /> All Devices in Organization<br />

<input type="submit" value="Show Devices" />

```

If the **configure_devices.php** script sets session variables for an array of one or more devices, an array of device classes, and an array of device templates, the **devices.php** page displays a table of devices:

```

<?php

if(isset($_SESSION['dev_list']) AND isset($_SESSION['class_list'])
AND count($_SESSION['dev_list']) > 0 AND isset($_SESSION
['templates'])) {

    $templates = $_SESSION['templates'];

```

The variable **\$table** is used to build the HTML that will display the table of devices. A foreach loop iterates through each device in the array of devices:

```

$table = "<table><tr><th>Device Name</th><th>Device
IP</th><th>Device Type</th><th>Service Level</th></tr>";

```



```
foreach($_SESSION['dev_list'] as $key => $device) {
```

On each iteration of the foreach loop, the **\$device** variable is set to an array of all fields for the current device. The **configure_devices.php** script sets the value of the custom field "c-last_dev_tpl" only if a template other than the base template is applied. Therefore, if "c-last_dev_tpl" is NULL, it is assumed that the base template was the last template to be applied to the device. The variable **\$service_level** is initialized to the last device template applied to the current device:

```
if(is_null($device['c-last_dev_tpl']) or $device['c-last_dev_
tpl'] == "") {
```

```
    $service_level = get_base_template();
```

```
}
```

```
else {
```

```
    $service_level = $device['c-last_dev_tpl'];
```

```
}
```

The **Device Name** and **Device IP** columns are populated using the appropriate values from the **\$device** array. The **Device Type** column is populated using values from the array of device classes (**\$_SESSION['class_list']**). The keys in the array of device classes are the device class URIs; the field from the device resource that references the associated device class (class_type) is used to look up the device class for this device. The class and description fields from the device class are combined to populate the **Device Type** column:

```
$table .= "<tr>";
```

```
$table .= "<td>" . $device['name'] . "</td>";
```

```
$table .= "<td>" . $device['ip'] . "</td>";
```

```
$table .= "<td>" . $_SESSION['class_list'][$device['class_
type']]['class'] . " " . $_SESSION['class_list'][$device['class_
type']]['description'] . "</td>";
```

The drop-down list in the **Service Level** column is constructed using the array of device templates (**\$templates**). The name of the drop-down list (which will appear as a key in the **\$_POST** array) is set to the URI of the current device (the current array key from **\$_SESSION['dev_list']**):

```
$table .= "<td><select name=\"\" . $key . \"\">";
```

A drop-down list option is added for each device template. The last template that was applied to the device (***\$service_level***) is selected as the default option. If the last template that was applied to the device is selected when the form is submitted, the value of the drop-down list is set to NULL so that it can be skipped by the ***configure_devices.php*** script:

```
foreach($templates as $template) {

    if($service_level == $template['URI']) {

        $table .= "<option value=NULL selected=\"selected\">" .
$template['description'] . "</option>";

    }

}
```

For all other device templates, the value of the drop-down list is set to the URI of the device template:

```
    else {

        $table .= "<option value=\"" . $template['URI'] . "\">" .
$template['description'] . "</option>";

    }

}

$table .= "</select>";

$table .= "</tr>";

}
```

The HTML for the table is completed and outputted. To prevent an error in the next execution of ***configure_devices.php*** from producing erroneous results in this page, the session variables that contain the array of devices, the array of device classes, and the array of device templates are unset:

```
$table .= "<tr><td colspan=\"3\"></td><td><input type=\"submit\"
name=\"config\" value=\"Configure Devices\" /></td></tr>";

$table .= "</table>";

echo $table;

unset($_SESSION['dev_list']);
```

```

unset($_SESSION['class_list']);

unset($_SESSION['templates']);

}

```

If an empty array of devices is returned by ***configure_devices.php***, an informational message is displayed instead of an empty table:

```

elseif(isset($_SESSION['dev_list'])) {

    echo "<p>No Devices Discovered</p>";

    unset($_SESSION['dev_list']);

    unset($_SESSION['class_list']);

}

?>

</form>

</body>

</html>

```

remove.php

The ***remove.php*** file provides a user interface for removing a customer from the system:

```

<html>

<head>

    <title>Remove Customer</title>

    <link href="provisioning.css" rel="stylesheet" type="text/css">

</head>

<body>

```

```

<?php

    require_once 'header.php';

?>

<form action="delete_customer.php" method="post">

    Customer Name:<br />

    <input type="text" name="customer" /><br />

    <input type="submit" value="Submit" />

</form>

</body>

</html>

```

When you enter a customer name and select the **[Submit]** button, the [delete_customer.php script](#) deletes all devices, credentials, and discovery sessions associated with that customer's organization record; deletes the organization record; and then returns a status message to remove.php.

provisioning.css

The **provisioning.css** file includes style information for the user interface pages. In this example, minimal style is applied to the user interface pages. You can customize the user interface pages by adding style information to this file:

```

table {width: 100%; border-collapse:collapse; text-align: center;}

th {border: solid 1px;}

td {border: solid 1px;}

```

Provisioning a Customer (provision_customer.php)

The **provision_customer.php** script processes the input values from **index.php** and performs the following provisioning tasks:

- If an organization record does not currently exist for the customer, creates an organization record for the customer.
- Ensures that SNMP credentials are configured for each supplied SNMP community string.
- Creates a discovery session for the customer using the configured SNMP credentials and the supplied list of IP addresses.
- Runs the discovery session.

If all of these tasks are successful, the script redirects to ***configure_devices.php***. ***configure_devices.php*** will return a list of discovered devices to the ***devices.php*** page. If a provisioning task is unsuccessful, ***provision_customer.php*** returns an error message to ***index.php***.

All back-end files:

- Use PHP session variables to return values to the user interface files.
- Use the functions defined in the ***utils.php*** file.
- Use the URL of an Administration Portal, Database Server, or All-In-One Appliance.

The ***provision_customer.php*** script starts by initializing the session, requiring ***utils.php***, and initializing ***\$base_uri*** to the URL of an Administration Portal, Database Server, or All-In-One Appliance:

```
<?php

session_start();

require_once 'utils.php';

$base_uri = get_admin_uri();
```

The ***provision_customer.php*** script validates the input to ensure that a customer name, IP address list, and either a community string list or the discover Non-SNMP flag were supplied:

```
if(isset($_POST['customer']) AND $_POST['customer'] != "" AND isset($_POST['ip_addresses']) AND $_POST['ip_addresses'] != "" AND

((isset($_POST['community_strings']) AND $_POST['community_strings'] != "") OR (isset($_POST['non_snmp']) AND $_POST['non_snmp'] == "yes"))) {
```

The ***provision_customer.php*** script attempts to lookup the URI of the organization record associated with the customer name supplied in the input form. If no organization record is found, the script creates a new organization record using the ***create_entity*** function. The array of fields for the new organization record includes only the name of the organization:

```
$organization = lookup_organization($base_uri, $_POST['customer']);

if($organization === FALSE) {
```

```

$org_post_array = array('company' => $_POST['customer']);

$org_response = create_entity($base_uri, "/api/organization", $org_post_array);

```

If the request to create an organization record is successful (the **create_entity** function returns TRUE at array index 0), the **\$organization** variable is set to the URI of the organization:

```

if($org_response[0]) {

    $organization = $org_response[1];

}

```

If the request to create an organization fails, the **\$message** variable is set to an appropriate error message:

```

else {

    $message = "Failed to create org: " . $org_response[1];

}

}

```

If an organization record already exists for the supplied customer name, the **provision_customer.php** script deletes any existing discovery sessions associated with that organization record. By deleting existing discovery sessions, the **provision_customer.php** script maintains a 1:1 mapping between organization records and discovery sessions. Maintaining a 1:1 mapping reduces the amount of processing required to retrieve a list of devices from the last discovery session that was run for a particular customer. The **provision_customer.php** script constructs a URI for the /discovery_session resource index that includes a filter for the organization record ID. The organization record ID is appended to the URI by using the last element in an array that contains each piece of the organization URI:

```

else {

    $uri = "/api/discovery_session?limit=10&hide_
filterinfo=1&filter.organization=" . array_pop(explode("/",
$organization));

    $response = perform_request($base_uri, $uri, "GET");

```

If the request for a list of discovery sessions is successful, the **multi_delete** function is called to delete the discovery sessions in the response. The **multi_delete** function returns NULL if all the supplied entities are deleted or an error message if one or more supplied entities are not deleted. If the request for the list of discovery sessions fails or if **multi_delete** did not return null, the **\$message** variable is set to an appropriate error message:

```
if($response['http_code'] == 200) {

    $error = multi_delete($base_uri, $response['content']);

}

else {

    $error = "Could not clean up existing discovery sessions for
organization.";

}

if(!is_null($error)) {

    $message = $error;

}

}
```

If no error message has been set in the **\$message** variable, the **provision_customer.php** script continues with the provisioning process:

```
if(!isset($message)) {
```

If a list of community strings was supplied in the input form, the **configure_credentials** function is used to get an array of credential URIs for those community strings:

```
if(isset($_POST['community_strings']) AND $_POST['community_
strings'] != "") {

    $credentials = configure_credentials($base_uri, $_POST
['customer'], $_POST['community_strings']);

}
```

If a list of community strings was not supplied in the input form, i.e. the discovery session will be configured to discover only non-SNMP devices, the **\$credentials** variable is initialized as an empty array:

```
else {  
  
    $credentials = array();  
  
}
```

If the **\$credentials** variable is an array, i.e. no error message was returned by the **configure_credentials** function or the discovery session will be configured to discover only non-SNMP devices, the script explodes the supplied list of IP addresses in to an array:

```
if(is_array($credentials)) {  
  
    $ip_array = explode(",", $_POST['ip_addresses']);
```

To create a discovery session using the API, the JSON content must include a list of IP address ranges. Each IP address range must specify a start address and an end address. In PHP array format, the array that contains the discovery session fields must include an "ip_lists" key that points to an array that has the following structure:

```
(  
  
    [0] => array (  
  
        ['start_ip'] =>  
  
        ['end_ip'] =>  
  
    )  
  
    .  
  
    .  
  
    .  
  
    [N] => array (  
  
        ['start_ip'] =>  
  
        ['end_ip'] =>  
  
    )
```



```
)
```

The script initializes the variable ***\$ip_lists***, which will contain this structure. For each IP address in the array of IP addresses supplied in the input form, an element is added to ***\$ip_lists***. Each IP address is used as both the start and end address for each IP address "range":

```
$ip_lists = array();

foreach($ip_array as $address) {

    $ip_lists[] = array('start_ip' => $address, 'end_ip' =>
        $address);

}
```

The script then uses the **get_collector_id** function to get the URI of an appliance on which the discovery session can run:

```
$collector = get_collector_id($base_uri);
```

The **get_collector_id** returns an array. The boolean value at array index 0 indicates whether an appliance URI was successfully returned. The value at array index 1 is either an appliance URI or an error message. If an appliance URI was returned, a discovery session is created using the following field values:

- ***organization***. The organization URI (***\$organization***).
- ***aligned_collector***. The appliance URI returned by the **get_collector_id** function.
- ***aligned_device_template***. The standard device template returned by the **get_base_template** function.
- ***initial_scan_level***. To limit what is monitored on each discovered device to only what is defined in the applied device templates, the initial scan level is set to 0 (*Model Device Only*).
- ***ip_lists***. The array of start and end IP addresses (***\$ip_lists***).
- ***credentials***. The array of credentials returned by the **configure_credentials** function.
- ***discover_non_snmp***. If **Discover Non-SNMP Devices** was selected in the input form, this value is set to 1 (discover non-SNMP devices).

```
if($collector[0]) {

    $disc_post_array = array('organization' => $organization,

        'aligned_collector' => $collector[1],
```

```

'aligned_device_template' => get_base_template(),

'initial_scan_level' => 0,

'ip_lists' => $ip_lists,

'credentials' => $credentials);

}

if(isset($_POST['non_snmp']) AND $_POST['non_snmp'] == "yes") {

    $disc_post_array['discover_non_snmp'] = 1;

}

$disc_response = create_entity($base_uri, "/api/discovery_
session", $disc_post_array);

```

The **create_entity** function returns an array. The boolean value at array index 0 indicates whether the entity was successfully created. The value at array index 1 is either the entity URI or an error message. If the discovery session was created successfully, the discovery session is started by applying the URI of the discovery session to the /discovery_session_active resource index:

```

if($disc_response[0]) {

    $run_discovery = perform_request($base_uri, "/api/discovery_
session_active", "APPLY", $disc_response[1]);

```

If the response from the request to start the discovery session includes HTTP status code 202, the discovery session started correctly. If the discovery session starts correctly, the script redirects to the **configure_devices.php** script. The **configure_devices.php** script requires the customer name as input, either in the \$_POST or \$_SESSION array. In this case, the customer name is set as a session variable:

```

if($run_discovery['http_code'] == 202) {

    $_SESSION['customer'] = $_POST['customer'];

    header("Location: configure_devices.php");

}

```

If the request to start the discovery session failed, i.e. the HTTP status code in the response is not 202, the ***\$message*** variable is set to an appropriate error message:

```
        else {  
  
            $message = "Failed to run discovery session: " . $run_  
discovery['http_code'];  
  
        }  
  
    }
```

If an error was returned by the **create_entity** function, the ***\$message*** variable is set to an appropriate error message:

```
        else {  
  
            $message = "Failed to create discovery session: " . $disc_  
response[1];  
  
        }  
  
    }
```

If an error was returned by the **get_collector_id** function, the ***\$message*** variable is set to an appropriate error message:

```
        else {  
  
            $message = $collector[1];  
  
        }  
  
    }
```

If ***\$credentials*** is not an array, i.e. the **configure_credentials** function returned an error message, the ***\$message*** variable is set to an appropriate error message:

```
        else {  
  
            $message = "Failed to configure credentials: " . $credentials;  
  
        }
```

```
}  
  
}
```

If the values supplied in the input form fail validation, the *\$message* variable is set to an appropriate error message:

```
else {  
  
    $message = "Form Incomplete";  
  
}
```

If the *\$message* variable is set, a failure occurred in the *provision_customer.php* script. The error is set in a session variable and the script redirects back to *index.php*:

```
if(isset($message)) {  
  
    $_SESSION['message'] = $message;  
  
    header("Location: index.php");  
  
}  
  
?>
```

Retrieving and Configuring Devices (configure_devices.php)

The *configure_devices.php* script returns a list of devices and associated device classes for a specified customer. The list of devices can be all devices associated with the customer's organization record, all devices from the last discovery session for that customer, or new devices from the last discovery session for that customer.

Additionally, if a user selects the **[Configure Devices]** button in the *devices.php* page, the *configure_devices.php* script applies the device templates selected by the user to the specified devices.

All back-end files:

- Use PHP session variables to return values to the user interface files.
- Use the functions defined in the *utils.php* file.
- Use the URL of an Administration Portal, Database Server, or All-In-One Appliance.

The script starts by initializing the session, requiring *utils.php*, and initializing *\$base_uri* to the URL of an Administration Portal, Database Server, or All-In-One Appliance:

```
<?php

session_start();

require_once 'utils.php';

$base_uri = get_admin_uri();
```

For each displayed device, the *devices.php* page displays a drop-down list that contains all device templates in the system. To populate the drop-down list, the *devices.php* page must be supplied a list of device templates. An array of all device templates is set as a session variable. Because the list of device templates is assumed to be static, the array of device templates is set only once per session and is never explicitly unset by the provisioning code. The code that creates the array of device templates is located in *configure_devices.php* because the script is always run before devices are displayed in *devices.php*.

If the *templates* variable is not currently set in the session variables, the script gets a list of all device templates using the *get_all* function. The *get_all* function returns an array of entities on success, or an error message on failure. If the return value is an array, that array is set as a session variable. If the return value is not an array, the *\$message* variable is assigned the returned error message:

```
if(!isset($_SESSION['templates'])) {

    $templates = get_all($base_uri, "/api/device_template?link_disp_
    field=template_name");

    if(is_array($templates)) {

        $_SESSION['templates'] = $templates;

    }

    else {

        $message = $templates;

    }

}
```

The ***configure_devices.php*** script takes a customer name as input. The customer name is passed either as post data from ***devices.php*** or in a session variable from ***provision_customer.php***. The script uses the customer name to lookup the organization URI:

```
if(isset($_POST['customer']) AND $_POST['customer'] != "") {  
  
    $customer = $_POST['customer'];  
  
    $_SESSION['customer'] = $_POST['customer'];  
  
    $organization = lookup_organization($base_uri, $customer);  
  
}  
  
elseif(isset($_SESSION['customer'])) {  
  
    $customer = $_SESSION['customer'];  
  
    $organization = lookup_organization($base_uri, $customer);  
  
}
```

If an organization URI is found for the supplied customer name, an error message has not been set, and the user selected the **[Configure Devices]** button ("config" is a key in the post data), the block of code that applies device templates to devices is executed:

```
if(array_key_exists("config", $_POST) AND $organization != FALSE AND  
!isset($message)) {
```

The variable ***\$dev_type*** is initialized with the value from the radio buttons on the ***devices.php*** page. Later in the execution of the ***configure_devices.php*** script, ***\$dev_type*** is used to set a session variable that the ***devices.php*** page uses as the default value of the radio buttons:

```
if(isset($_POST['dev_type'])) {  
  
    $dev_type = $_POST['dev_type'];  
  
}
```

The script iterates through all values supplied by the input form. The variable ***\$devices_updated*** is initialized to track the number of devices to which device templates are applied. The variable ***\$in*** is

initialized as an array, which will be used to track the device ID values for all devices that were previously displayed on *devices.php*. The *\$in* array will be used to return the same list of devices to *devices.php*:

```
$in = array();
```

```
$devices_updated = 0;
```

```
foreach($_POST as $device => $template) {
```

The *\$_POST* array includes all values supplied by the input form. This block of code must operate only on the values from the drop-down list for each device. Each drop-down list is named using the URI of the associated device; therefore, it is assumed that if a key in the *\$_POST* array begins with a slash character ("/"), the array element represents a drop-down list:

```
if(strpos($device, "/") === 0) {
```

The device ID of all devices that were displayed on *devices.php* is added to the *\$in* array:

```
$in[] = array_pop(explode("/", $device));
```

If the user did not select a new device template from the drop-down list for a device, the value for that drop-down list is "NULL". The block of code that applies a device template to a device is executed only if the value for the drop-down list is not "NULL". Note that the input form passes "NULL" as a string, not the NULL data-type:

```
if($template != "NULL") {
```

To apply a device template to a device, the script uses the *perform_request* function with a *\$type* parameter of "APPLY":

```
$apply_template = perform_request($base_uri, $device, "APPLY",  
$template);
```

If the request to apply a device template to a device is successful (the response includes a HTTP status code of 200), the script must update the device resource with the new value for the *c-last_dev_tpl* field. To do this, the device resource is requested:

```
if($apply_template['http_code'] == 200) {
```

```
$dev = perform_request($base_uri, $device, "GET");
```

If the request for the device resource is successful, the new value of "c-last_dev_tpl" field is set in the array of attributes for that device and the array of attributes is POSTed back to the same device resource:

```
if($dev['http_code'] == 200) {  
  
    $dev['content']['c-last_dev_tpl'] = $template;  
  
  
    $update_device = perform_request($base_uri, $device, "POST",  
$dev['content']);  
}
```

If the request to update a device resource fails, the variable ***\$message*** is initialized with an error message:

```
if($update_device['http_code'] != 200) {  
  
    $message = "Could not update template status of " .  
$device . ". ";  
  
    if(array_key_exists("error", $update_device)) {  
  
        $message .= $update_device['error'];  
  
    }  
  
}
```

If the request to update a device resource is successful, the number of devices that have been updated is incremented:

```
else {  
  
    $devices_updated++;  
  
}
```

If the request for a device resource fails, the variable ***\$message*** is initialized with an error message:

```
else {
```



```

        $message = "Could not get information to update template
status of " . $device . ". ";

```

```

        if(array_key_exists("error", $udev)) {

```

```

            $message .= $dev['error'];

```

```

        }

```

```

    }

```

```

}

```

If the request to apply a device template to a device fails, the variable ***\$message*** is initialized with an error message:

```

    else {

```

```

        $message = "Could not apply " . $template . " to " . $device .
". ";

```

```

        if(array_key_exists("error", $apply_template)) {

```

```

            $message .= $apply_template['error'];

```

```

        }

```

```

    }

```

```

}

```

```

}

```

```

}

```

The ***\$in*** array, which includes the device IDs of all devices that were previously displayed in ***devices.php***, is used to re-request the list of devices:

```

$uri = "/api/device?extended_fetch=1&filter.id.in=" . implode(",",
$in);

```

```

$device_list = get_all($base_uri, $uri);

```

If the request for the list of devices is successful, the script requests a list of device classes for those devices:

```
if(is_array($device_list)) {  
  
    $class_list = get_join_resources($base_uri, $device_list, "class_  
type", "class_type");
```

If the request for a list of device classes is successful, the array of devices and the array of device classes are passed back to **devices.php** in session variables:

```
if(is_array($class_list)) {  
  
    $_SESSION['class_list'] = $class_list;  
  
    $_SESSION['dev_list'] = $device_list;  
  
}
```

If the request for the list of device classes fails, the variable **\$message** is initialized with the error message returned by the **get_join_resources** function:

```
else {  
  
    $message = $class_list;  
  
}  
  
}
```

If the request for the list of all devices fails, the variable **\$message** is initialized with the error message returned by the **get_all** function:

```
else {  
  
    $message = $device_list;  
  
}
```

If the **\$message** variable has not yet been initialized, all requests were successful and the **\$message** variable is initialized with a success message:

```
if(!isset($message)) {
```

```

        $message = $devices_updated . " Device(s) Updated.";
    }
}

```

If the block of code that configures devices is not executed, but an organization URI has been found for the customer and an error message has not been set, the block of code that returns a list of devices is executed:

```
elseif($organization != FALSE AND !isset($message)) {
```

The variable ***\$org_id*** is initialized with the ID of the organization for which a list of devices has been requested:

```
$org_id = array_pop(explode("/", $organization));
```

If a value has been supplied from the radio buttons on the ***devcies.php*** page, the variable ***\$dev_type*** is initialized with that value. If no value has been supplied, e.g. the script was called by ***provision_customer.php***, the script defaults to returning a list of new devices from the last discovery session:

```

if(isset($_POST['dev_type'])) {

    $dev_type = $_POST['dev_type'];

}

else {

    $dev_type = "new_disc";

}

```

The following block of code returns a list of all devices associated with the specified organization:

```
if($dev_type == "all_org") {
```

The organization ID is used as the filter criteria to request a list of devices. The ***get_all*** function will return an array of all devices that match the filter criteria:

```
$uri = "/api/device?extended_fetch=1&filter.organization=" . $org_id;
```

```
$device_list = get_all($base_uri, $uri);
```

If the request for a list of devices is successful, the **get_all** function returns an array and the script requests a list of device classes for those devices:

```
if(is_array($device_list)) {
```

```
    $class_list = get_join_resources($base_uri, $device_list, "class_type", "class_type");
```

If the request for a list of device classes is successful, the array of devices and the array of device classes are passed back to `devices.php` in session variables:

```
if(is_array($class_list)) {
```

```
    $_SESSION['class_list'] = $class_list;
```

```
    $_SESSION['dev_list'] = $device_list;
```

```
}
```

If the request for the list of device classes fails, the variable **\$message** is initialized with the error message returned by the **get_join_resources** function:

```
else {
```

```
    $message = $class_list;
```

```
}
```

```
}
```

If the request for the list of devices fails, the variable **\$message** is initialized with the error message returned by the **get_all** function:

```
else {
```

```
    $message = $device_list;
```

```
}
}
}
```

If the block of code that returns a list of all devices associated with the specified organization is not executed, the script executes a block of code that returns a list of devices from the last discovery session:

```
else {
```

The organization ID is used as the filter criteria to request the discovery session for the customer:

```
$discovery_search = perform_request($base_uri, "/api/discovery_
session?limit=100&hide_filterinfo=1&filter.organization=" . $org_id,
"GET");
```

If the request for a discovery session is successful (the response includes HTTP status code 200) and at least one discovery session is returned, the script calls the **get_discovery_result** function using the first discovery session in the response. It is assumed that there is a 1:1 mapping between organizations and customers; a 1:1 mapping is maintained by the *provision_customer.php* script. The third parameter passed to **get_discovery_result** is a boolean that determines whether the function will return all devices discovered by the discovery session or only new devices discovered by the discovery session:

```
if($discovery_search['http_code'] == 200 AND count($discovery_search
['content']) > 0) {

    $device_list = get_discovery_result($base_uri, $discovery_search
['content'][0]['URI'], ($dev_type == "new_disc"));
```

The status code of the **get_discovery_result** is used in a switch statement that performs the required actions of each possible result:

```
switch($device_list['status']) {
```

A status code of 1 indicates that a list of devices has been returned, but the discovery session is still running. If the discovery session is still running, the *\$message* variable is initialized to an appropriate status message. The required actions for a status code of 0 must also be performed if the status code is 1, so no break statement is included for case 1:

```
case 1:
```

```
    $message = "Note: Discovery Session is not complete,  
additional devices might be discovered.";
```

A status code of 0 indicates that a list of devices has been returned and the discovery session is complete. If the array of devices is not empty, the **\$in** variable is initialized as an array that will be used to track the device IDs of all devices in the array of devices:

```
case 0:
```

```
    if(count($device_list['devices']) > 0) {
```

```
        $in = array();
```

The script iterates through the array of devices. For each device, the device ID is derived from the device URI and is added to the **\$in** array:

```
        foreach($device_list['devices'] as $device) {
```

```
            $in[] = array_pop(explode("/", $device['uri']));
```

```
        }
```

A URI to request all devices is constructed using the device IDs in the **\$in** array. The **get_all** function is used to request all the devices:

```
        $uri = "/api/device?extended_fetch=1&filter._id.in=" .  
implode(",", $in);
```

```
        $device_list = get_all($base_uri, $uri);
```

If the request for the list of devices is successful, the script requests a list of device classes for those devices:

```
        if(is_array($device_list)) {
```

```
            $class_list = get_join_resources($base_uri, $device_list,  
            "class_type", "class_type");
```

If the request for a list of device classes is successful, the array of devices and the array of device classes are passed back to **devices.php** in session variables:

```
if(is_array($class_list)) {  
  
    $_SESSION['class_list'] = $class_list;  
  
    $_SESSION['dev_list'] = $device_list;  
  
}
```

If the request for the list of device classes fails, the variable **\$message** is initialized with the error message returned by the **get_join_resources** function:

```
else {  
  
    $message = $class_list;  
  
}  
  
}
```

If the request for the list of devices fails, the variable **\$message** is initialized with the error message returned by the **get_all** function:

```
else {  
  
    $message = $device_list;  
  
}  
  
}
```

If the **get_discovery_result** function returned an empty array of devices, an empty array of devices is passed to **devices.php** in a session variable:

```
else {  
  
    $_SESSION['dev_list'] = array();  
  
}  
  
break;
```

If the **get_discovery_result** returns status 2 (discovery session has never been run), status 3 (error), or a status other than 0, 1, 2, or 3, script sets the **\$message** variable to an appropriate error message:

```
        case 2:

            $message = $device_list['error'];

            break;

        case 3:

            $message = $device_list['error'];

            break;

        default:

            $message = "Error occurred retrieving discovery logs.";

    }

}
```

If the request for a discovery session returned an HTTP status code of 200 but did not return any discovery sessions, the script sets the **\$message** variable to an appropriate error message:

```
elseif($discovery_search['http_code'] == 200) {

    $message = "No discovery session exists for customer.";

}
```

If the request for a discovery session returned an HTTP status code other than 200, the script sets the **\$message** variable to an appropriate error message:

```
else {

    $message = "Error finding discovery session for customer. ";

    if(array_key_exists("error", $discovery_search)) {

        $message .= $discovery_search['error'];

    }

}
```



```

    }

}

}

}

```

If no organization URI was found for the supplied customer name, the script sets the ***\$message*** variable to an appropriate error message:

```

elseif(!isset($message)) {

    $message = "Could not find customer record." ;

}

```

If the ***\$message*** and/or ***\$dev_type*** variables have been set during the execution of the script, they are passed to ***devices.php*** using session variables. The script always redirects back to ***devices.php***:

```

if(isset($message)) {

    $_SESSION['message'] = $message;

}

if(isset($dev_type)) {

    $_SESSION['dev_type'] = $dev_type;

}

header("Location: devices.php");

?>

```

Removing a Customer (delete_customer.php)

The ***delete_customer.php*** script takes a customer name as input; deletes all devices, credentials, and discovery sessions associated with that customer's organization record; and then deletes the organization record for that customer.

All back-end files:

- Use PHP session variables to return values to the user interface files.
- Use the functions defined in the *utils.php* file.
- Use the URL of an Administration Portal, Database Server, or All-In-One Appliance.

The script starts by initializing the session, requiring *utils.php*, and initializing *\$base_uri* to the URL of an Administration Portal, Database Server, or All-In-One Appliance:

```
<?php

session_start();

require_once 'utils.php';

$base_uri = get_admin_uri();
```

If a customer name was supplied in the input form, the script looks up the URI for the organization record associated with that customer name. If no organization record is found, the *\$message* variable is set to an error message:

```
if(isset($_POST['customer']) AND $_POST['customer'] != "") {

    $organization = lookup_organization($base_uri, $_POST['customer']);

    if($organization === FALSE) {

        $message = "Organization does not exist";

    }
```

If an organization record exists for the customer, the ID for that organization record is parsed from the URI:

```
else {

    $org_id = array_pop(explode("/", $organization));
```

An array of resource index URIs is constructed. The script will iterate through this array and delete all entities returned by each URI. The organization ID is used as filter criteria in each URI. If a request fails, the array keys are used to indicate the entity type where the problem occurred. To delete additional entities associated with the customer organization, for example, asset records, you can add additional URIs to this array:

```

$entity_types = array("devices" => "/api/device?limit=100&hide_
filterinfo=1&filter.organization=" . $org_id,

"credentials" => "/api/credential/snmp?limit=100&hide_
filterinfo=1&filter.cred_name.contains=" . $_POST['customer'],

"discoveries" => "/api/discovery_session?limit=100&hide_
filterinfo=1&filter.organization=" . $org_id,

);

```

The script iterates through the array of URIs. For each URI:

- A GET request is performed.
- If the GET request is successful, the content in the response is passed to the **multi_delete** function, which will delete all the returned entities.
- If the GET request is unsuccessful, an error message is set in the **\$error** variable.
- The steps are repeated until either an error occurs or the GET request returns no entities.

If an error occurs for a URI, the iteration through the array of URIs stops:

```

foreach($entity_types as $key => $entity) {

    do {

        $response = perform_request($base_uri, $entity, "GET");

        if($response['http_code'] == 200) {

            $error = multi_delete($base_uri, $response['content']);

        }

        else {

            $error = "Could not get list of " . $key . " to delete";

        }

    } while(count($response['content']) > 0 AND $error == NULL);

    if(!is_null($error)) {

        break;
    }
}

```

```
}
}
```

To delete an organization record from Skylar One, the organization must be "empty", that is, have no entities associated with it. If no error was generated when the other entities were deleted, the organization is deleted:

```
if(is_null($error)) {

    $response = perform_request($base_uri, $organization, "DELETE");

    if($response['http_code'] == 200) {

        $message = "Customer removed";

    }
}
```

If the request to delete the organization failed with a 400 HTTP status code, the organization is not empty and an error message is set in the **\$message** variable:

```
elseif($response['http_code'] == 400) {

    $message = "Could not delete organization because organization
    is not empty.";

}
```

If the request to delete the organization failed with a different HTTP status code, a generic error message is set in the **\$message** variable:

```
else {

    $message = "Could not delete organization. ";

    if(array_key_exists("error", $response)) {

        $message .= $response['error'];

    }

}
```

```
}
```

If the ***\$error*** variable has already been set because deleting an entity other than the organization failed, ***\$message*** is set to the value of ***\$error***.

```
else {
```

```
    $message = $error;
```

```
}
```

```
}
```

```
}
```

If no customer name is specified in the input form, an appropriate error message is set in the ***\$message*** variable:

```
else {
```

```
    $message = "Form Incomplete";
```

```
}
```

If the ***\$message*** variable has been set, its value is returned to remove.php using a session variable:

```
if(isset($message)) {
```

```
    $_SESSION['message'] = $message;
```

```
}
```

```
header("Location: remove.php");
```

```
?>
```

Create Device Maintenance Schedules via the API

Overview

This chapter describes how to create device maintenance schedules using the API.

This chapter covers the following topics:

<i>Requirements</i>	263
<i>Prerequisite Examples</i>	263
<i>Getting Started</i>	265
<i>Creating the Task (Step 1)</i>	265
<i>Creating the Schedule Entry (Step 2)</i>	267
<i>Aligning the Task to the Schedule Entry (Step 3)</i>	269

Requirements

To successfully create a device maintenance schedule via the ScienceLogic API, you need three API calls that do the following:

- Create the task.
- Create the schedule.
- Align the task to the schedule.

Caveats to Consider

- Data will not populate into the Skylar One user interface until the third alignment call is complete.
- The schedule will appear in three places in Skylar One:
 - The **Schedule Manager** page (Registry > Schedules > Schedule Manager)
 - The **[Schedules]** tab of the **Device Investigator** (Devices > select a device > **[Schedules]**)
 - The **[Schedule]** tab of the **Device Properties** page (Devices > Classic Devices > wrench icon () > Schedule)
- The return of the `task_id` and `schedule_id` values from each creation call will vary. Values used here are examples only.

Prerequisite Examples

To create a device maintenance schedule, you must first collect the variable values you will use. The following table describes those variables required for the API calls and provides example variables.

Variable Name	Values Used as Examples	Description
USERNAME	em7admin	The username used to authenticate the API request
PASSWORD	password	The password associated with the username used to authenticate the API request
IP-ADDRESS-OF-DB-APPLIANCE	10.64.70.29	The IP address of your Database Server
NAME	SCHEDULETEST-API	Sets the name of the task
DESCRIPTION	Create Device Maintenance Schedule via API	Sets a brief overview of the task purpose

ENABLED VALUE	1	States whether the task is enabled (1) or disabled (2).
ACTIVE WINDOW	12	Sets whether collection will continue during the window (4) or if it will be disabled (12)
PATCH WINDOW VALUE	0	Sets whether this maintenance window coincides with a patch window or not. If yes, set this value to the number of minutes in the patch window (5 to 60, in increments of 5 minutes); otherwise, set to 0.
OWNER ACCOUNT VALUE	1	The user ID of the user responsible for this task. Can be retrieved from the User Accounts page (Registry > Accounts > User Accounts).
DEVICE ORG VALUE	1	The organization ID to which the device belongs. Can be retrieved from the Organizational Account Administration page (Registry > Accounts > Organizations).
VISIBILITY VALUE	Organization	Sets the visibility level of this maintenance schedule. Can be set to: • Private. Visible only to the owner. • Organization. Visible to anyone in the organization. • World. Visible to everyone.
ALIGNED RESOURCE	2	The ID of the device that will be placed in maintenance mode at the designated time
DATE START	1725983558	Sets the date/time in Epoch format for when the maintenance window begins
TIMEZONE VALUE	157	Sets the timezone to which the user interface will convert the time from UTC when displaying. The ID number can be retrieved from the master.definitions_timezones table in the Skylar One database.
DURATION	60	Sets the length of the maintenance window, in minutes
RECUR_EXPR_VALUE and RECUR_UNIT_VALUE	0 and null	Sets the schedule recurrence interval. The "Expr" value is a number, and the Unit is a unit of time: "MINUTE", "HOUR", "DAY", "WEEK" or "MONTH". For a one-time schedule, set the RECUR_UNIT_VALUE to null and the RECUR_EXPR_VALUE to "0".

RECUR_UNTIL _ VALUE	0	Sets the end date of a recurring schedule. Must be set in Epoch time, if used. For a one-time schedule, set the RECUR_UNTIL _VALUE to "0"
TASK VALUE	40	The ID of the task created in Step 1
SCHEDULE VALUE	38	The ID of the schedule created in Step 2

Getting Started

In this example, your device ID 1 will be placed into a 30-minute maintenance window. Replace the following:

- Username: "USERNAME" - your API user
- Password: "PASSWORD" - your API user password
- IP Address: "IP-ADDRESS-OF-DB-APPLIANCE" - your platform IP address or URI for API connection

You can access the database to review the information after each API call to verify successful creation.

NOTE: The **Database Tool** page (System > Tools > DB Tool) is available only in versions of Skylar One prior to 12.2.1 and displays only for users that have sufficient permissions to access the page. For more information, see this [Knowledge Base article](#).

- scheduler.tasks

```
SELECT * FROM scheduler.tasks WHERE task_id = {task_id created in API call}
```

- scheduler.schedules

```
SELECT * FROM scheduler.schedules WHERE schedule_id = {schedule_id created in API call}
```

- scheduler.schedules_to_tasks

This aligns the tasks to the schedules, without which the return of information will not populate into the user interface:

```
SELECT * FROM scheduler.schedules_to_tasks WHERE schedule_id = {schedule_id created in API call} AND task_id = {task_id created in API call}
```

Creating the Task (Step 1)

The following is an example of how to create the task in an API call using the data in the table above. The command in this example requires you to replace the placeholders in square brackets with your data.

```
curl -sku '[USERNAME]:[PASSWORD]' -H 'content-type:application/json; charset=UTF-8' -X POST https://[IP-ADDRESS-OF-DB-APPLIANCE]/api/task -d '{
"name":"[NAME]","description":"[DESCRIPTION]","enabled":"[ENABLED VALUE]","details":{"\"active\":[ACTIVE VALUE],\"patch_window\":[PATCH WINDOW VALUE]}","last_run":null,"owner":"\\api\\account\\[OWNER VALUE]","organization":"\\api\\organization\\[DEVICE ORG VALUE]","visibility":"[VISIBILITY VALUE]","aligned_resource":"\\api\\device\\[ALIGNED RESOURCE]"}'
```

For example, if you enter the data from the **Values Uses as Examples** column in the table located in the [Prerequisite Examples](#) section, you will receive the following example response:

```
# curl -sku 'em7admin:password' -H 'content-type:application/json; charset=UTF-8' -X POST 'https://10.64.70.29/api/task'; -d '{
"name":"SCHEDULETEST-API","description":"Create Device Maintenance Schedule via API","enabled":"1","details":{"\"active\":12,\"patch_window\":0},"last_run":null,"owner":"\\api\\account\\1","organization":"\\api\\organization\\1","visibility":"Organization","aligned_resource":"\\api\\device\\2"}'
```

```
{
  "name": "SCHEDULETEST-API",
  "description": "Create Device Maintenance Schedule via API",
  "enabled": "1",
  "details": {
    "\"active\"": 12,
    "\"patch_window\"": 0
  },
  "last_run": null,
  "owner": "\\api\\account\\1",
  "organization": "\\api\\organization\\1",
  "visibility": "Organization",
  "task_guid": "E41CC650F3488BE76D1245C7F355583D",
  "schedules": [
    {
      "URI": "\\api\\task\\40\\schedule?limit=100",
      "description": "Index of schedules associated with this task"
    }
  ],
  "aligned_resource": "\\api\\device\\2"
}
```

Make sure to take note of the task number returned in the **URI** field. In this example, the value is 40.

To verify that you successfully created the task, you can query the database.

NOTE: The **Database Tool** (System > Tools > DB Tool) is available only in versions of Skylar One prior to 12.2.1 and displays only for users that have sufficient permissions to access the page. For more information, see this [Knowledge Base article](#).

```
# silo_mysql -e "SELECT * FROM scheduler.tasks WHERE task_id = 40 \G"
***** 1. row *****
task_id: 40
context: 2
```

```

    name: SCHEDULETEST-API
description: Create Device Maintenance Schedule via API
    Xid: 2
    enabled: 1
    details: {"active":12,"patch_window":0}
    last_run: NULL
    owner: 1
    roa_id: 1
visibility: Organization
    task_guid: E41CC650F3488BE76D1245C7F355583D

```

IMPORTANT: ScienceLogic recommends that SaaS customers on Skylar One version 12.2.1 or later review this [Knowledge Base article](#) for the current process to troubleshoot results of database queries and to submit a case to the proper team with the queries for which you want to see results.

Creating the Schedule Entry (Step 2)

The following command will create the device maintenance schedule entry and will post the information in the API in JSON format. As in [Step 1](#), the command in this example requires you to replace the placeholders in square brackets with your data.

```

curl -sku '[USERNAME]:[PASSWORD]' -H 'content-type:application/json;
charset=UTF-8' -X POST https://[IP-ADDRESS-OF-DB-APPLIANCE]/api/schedule -
d '{"dtstart":"[DATE START]","timezone":"[TIMEZONE VALUE]","duration":"
[DURATION]","description":"[DESCRIPTION]","recur_expr":"[RECUR_EXPR_
VALUE]","recur_unit":[RECUR_UNIT_VALUE],"recur_until":"[RECUR_UNTIL_
VALUE]","owner":"\\api\\account\\[OWNER ACCOUNT
VALUE]","organization":"\\api\\organization\\[DEVICE ORG
VALUE]","visibility":"[VISIBILITY VALUE]","tasks":[[TASK VALUE]]}'

```

For example, if you enter the data from the **Values Uses as Examples** column in the table located in the [Prerequisite Examples](#) section, you will receive the following example response:

```

# curl -sku 'em7admin:password' -H 'content-type:application/json;
charset=UTF-8' -X POST https://10.64.70.29/api/schedule -d '
{"dtstart":"1725983558","timezone":"157","duration":"60","description":"C
reate Device Maintenance Schedule via API","recur_expr":"0","recur_
unit":null,"recur_
until":"0","owner":"\\api\\account\\1","organization":"\\api\\organizatio
n\\1","visibility":"Organization","tasks":[40]}'

```

```
{
  "dtstart": "1725983558",
  "timezone": "157",
  "duration": "60",
  "description": "Create Device Maintenance Schedule via API",
  "recur_expr": "0",
  "recur_unit": null,
  "recur_until": "0",
  "owner": "\\api\\account\\1",
  "organization": "\\api\\organization\\1",
  "visibility": "Organization",
  "in_window": "0",
  "preserve_in_db": "0",
  "ppguid": null,
  "schedule_guid": "411854E24203FA8C29A8217B02881C86",
  "tasks": [
    {
      "URI": "\\api\\schedule\\38\\task?limit=100",
      "description": "Index of tasks associated with this schedule"
    }
  ]
}
```

Make sure to take note of the schedule number returned in the **URI** field. In this example, the value is 38. To verify that you successfully created the schedule entry, you can query the database.

NOTE: The **Database Tool** (System > Tools > DB Tool) is available only in versions of Skylar One prior to 12.2.1 and displays only for users that have sufficient permissions to access the page. For more information, see this [Knowledge Base article](#).

```
# silo_mysql -e "select * from scheduler.schedules where schedule_id = 38\G"
***** 1. row *****
schedule_id: 38
dtstart: 2024-09-10 15:52:38
timezone: 157
duration: 60
description: Create Device Maintenance Schedule via API
recur_expr: 0
recur_unit: NULL
recur_until: 0000-00-00 00:00:00
owner: 1
roa_id: 1
visibility: Organization
in_window: 0
preserve_in_db: 0
ppguid: NULL
schedule_guid: 411854E24203FA8C29A8217B02881C86
```

IMPORTANT: ScienceLogic recommends that SaaS customers on Skylar One version 12.2.1 or later review this [Knowledge Base article](#) for the current process to troubleshoot results of database queries and to submit a case to the proper team with the queries for which you want to see results.

Aligning the Task to the Schedule Entry (Step 3)

To align the task to the schedule entry, enter the following command. As in previous steps, the command in this example requires you to replace the placeholders in square brackets with your data.

```
curl -sku '[USERNAME]:[PASSWORD]' -H 'content-type:application/json; charset=UTF-8' -X POST https://[IP-ADDRESS-OF-DB-APPLIANCE]/api/schedule/[SCHEDULE VALUE] -d '{"tasks":[[TASK VALUE]]}'
```

Running this command applies the task ID (40) created in [Step 1](#) to the schedule entry (38) created in [Step 2](#).

For example, if you enter the data from the **Values Uses as Examples** column in the table located in the [Prerequisite Examples](#) section, you will receive the following example response:

```
# curl -sku 'em7admin:password' -H 'content-type:application/json; charset=UTF-8' -X POST https://10.64.70.29/api/schedule/38 -d '{"tasks": [40]}'

{"dtstart":"1725983558","timezone":"157","duration":"60","description":"Create Device Maintenance Schedule via API","recur_expr":"0","recur_unit":null,"recur_until":"0","owner":"\\api\\account\\1","organization":"\\api\\organization\\1","visibility":"Organization","in_window":"0","preserve_in_db":"0","ppguid":null,"schedule_guid":"411854E24203FA8C29A8217B02881C86","tasks":{"URI":"\\api\\schedule\\38\\task?limit=100","description":"Index of tasks associated with this schedule"}}
```

To verify that you successfully aligned the task to the schedule entry, you can query the database.

NOTE: The **Database Tool** (System > Tools > DB Tool) is available only in versions of Skylar One prior to 12.2.1 and displays only for users that have sufficient permissions to access the page. For more information, see this [Knowledge Base article](#).

```
# silo_mysql -e "select * from scheduler.schedules_to_tasks where sched-  
ule_id = 38\G"  
***** 1. row *****  
schedule_id: 38  
task_id: 40
```

IMPORTANT: ScienceLogic recommends that SaaS customers on Skylar One version 12.2.1 or later review this [Knowledge Base article](#) for the current process to troubleshoot results of database queries and to submit a case to the proper team with the queries for which you want to see results.

You are also able to verify that you successfully completed the steps in the Skylar One user interface by reviewing the schedule in the following locations:

- The **[Schedules]** tab of the **Device Investigator** (Devices > select a device > **[Schedules]** tab)
- The **[Schedule]** tab of the **Device Properties** page (Devices > Classic Devices > wrench icon (🔧) > Schedule)
- The **Schedule Manager** page (Registry > Schedules > Schedule Manager)

Appendix

A

Available Actions

Overview

This appendix lists all actions that can be performed through the API, organized by ScienceLogic entity. Each table includes the HTTP method and URI you should use to perform the action. The URIs in this list include "X", which signifies where the ID number of a specific entity must be inserted.

This chapter covers the following topics:

<i>Accounts</i>	274
<i>Account Lockouts</i>	274
<i>Alerts</i>	275
<i>Appliances</i>	275
<i>Assets</i>	275
<i>CBQoS Metrics</i>	278
<i>CBQoS Objects</i>	278
<i>CBQoS Object Types</i>	278
<i>Cleared Events</i>	278
<i>Collection Labels</i>	279
<i>Collection Label Groups</i>	279
<i>Collector Groups</i>	279
<i>Credentials</i>	280

Custom Attributes	282
Dashboards	284
Devices	285
Device Categories	288
Device Classes	288
Device Groups	288
Device Relationships	289
Device Relationship Types	289
Device Templates	290
Discovery Sessions	292
Dynamic Applications	293
Events	308
Event Categories	308
External Contacts	308
File Uploads	309
Interfaces	309
Interface Metrics	310
Interface Tags	310
Monitors	311
Organizations	313
Performance Data	314
PowerPacks	316
Product SKUs	316
Scale Values	317
Schedules	317
Streamer Push Proxy	317
System Patches	318
System Settings	319
System Thresholds	319
Tasks	319
Themes	320

<i>Threshold Overrides</i>	320
<i>Tickets</i>	321
<i>Ticket Categories</i>	322
<i>Ticket Chargeback</i>	323
<i>Ticket Logs</i>	323
<i>Ticket Notes</i>	323
<i>Ticket Queues</i>	324
<i>Ticket States</i>	324
<i>Unit Values</i>	325
<i>User Policies</i>	325
<i>Vendors</i>	325

Accounts

Action	URI	Method
View/search/filter the list of user accounts.	/account	GET
Create a new user account.	/account	POST
View the properties of a user account.	/account/X	GET
Update the properties of a user account.	/account/X	POST
Replace a user account.	/account/X	PUT
Delete a user account.	/account/X	DELETE
View the list of access hooks that have been granted to a user account.	/account/X/access_hooks	GET
For records that require an account value, use the user ID for the logged-in user.	/account/_self	GET, POST

IMPORTANT: Users with minimum permissions (for example, those with zero keys aligned to their accounts) can still view the IDs and names of other user accounts in their organization using the /account API endpoint.

Account Lockouts

Action	URI	Method
View a list of locked-out user accounts.	/access_lock	GET
View details about a locked-out user account.	/access_lock/X	GET
Clear a lock on a user account.	/access_lock/X	DELETE

Alerts

Action	URI	Method
Create a new API alert.	/alert	POST
View/search/filter the list of pending API alerts.	/alert	GET
View details about a pending API alert.	/alert/X	GET
Update a pending API alert.	/alert/X	POST

Appliances

Action	URI	Method
View/search/filter the list of Skylar One appliances.	/appliance	GET
View the properties of a Skylar One appliance.	/appliance/X	GET
Update the description or IP address of a Skylar One appliance.	/appliance/X	POST

Assets

Action	URI	Method
View/search/filter the list of asset records.	/asset	GET
Create a new asset record.	/asset	POST
View the general properties of an asset record.	/asset/X	GET
Replace an asset record.	/asset/X	PUT

Action	URI	Method
Update the general properties of an asset record.	/asset/X	POST
Delete an asset record.	/asset/X	DELETE
View/search/filter the list of components associated with an asset record.	/asset/X/component/	GET
Add a new component to an asset record.	/asset/X/component/	POST
View the properties of a component associated with an asset record.	/asset/X/component/X	GET
Update the properties of a component associated with an asset record.	/asset/X/component/X	POST
Replace a component associated with an asset record.	/asset/X/component/X	PUT
Delete a component from an asset record.	/asset/X/component/X	DELETE
View the configuration properties of an asset record.	/asset/X/configuration/	GET
Update the configuration properties of an asset record.	/asset/X/configuration/	POST
Replace the configuration properties of an asset record.	/asset/X/configuration/	PUT
View/search/filter the list of software licenses associated with an asset record.	/asset/X/license/	GET
Add a new software license to an asset record.	/asset/X/license/	POST
View the properties of a software license associated with an asset record.	/asset/X/license/X	GET
Update the properties of a software license associated with an asset record.	/asset/X/license/X	POST
Replace a software license associated with an asset record.	/asset/X/license/X	PUT
Delete a software license from an asset record.	/asset/X/license/X	DELETE

Action	URI	Method
View the maintenance and service properties of an asset record.	/asset/X/maintenance/	GET
Update the maintenance and service properties of an asset record.	/asset/X/maintenance/	POST
Replace the maintenance and service properties of an asset record.	/asset/X/maintenance/	PUT
View/search/filter the list of IP networks associated with an asset record.	/asset/X/network/	GET
Add a new IP network to an asset record.	/asset/X/network/	POST
View the properties of an IP network associated with an asset record.	/asset/X/network/X	GET
Update the properties of an IP network associated with an asset record.	/asset/X/network/X	POST
Replace an IP network associated with an asset record.	/asset/X/network/X	PUT
Delete an IP network from an asset record.	/asset/X/network/X	DELETE
View/search/filter the list of notes associated with an asset record.	/asset/X/note/	GET
Add a note to an asset record.	/asset/X/note/	POST
View a note associated with an asset record.	/asset/X/note/X	GET
Update a note associated with an asset record.	/asset/X/note/X	POST
Replace a note associated with an asset record.	/asset/X/note/X	PUT
View/search/filter the list of files associated with an asset record note.	/asset/X/note/X/media	GET
Get a media file associated with an asset record note.	/asset/X/note/X/media/X	GET
Add a media file to an asset record note.	/asset/X/note/X/media/X	PUT
View meta-data about a media file associated with an asset record note.	/asset/X/note/X/media/X/info	GET

CBQoS Metrics

Action	URI	Method
View/search/filter the list of CBQoS metrics.	/cbqos_metric	GET
View details about a CBQoS metric.	/cbqos_metric/X	GET

CBQoS Objects

Action	URI	Method
View/search/filter the list of CBQoS objects.	/cbqos_object	GET
View details about a CBQoS object.	/cbqos_object/X	GET

CBQoS Object Types

Action	URI	Method
View/search/filter the list of CBQoS object types.	/cbqos_type	GET
View details about a CBQoS object type.	/cbqos_type/X	GET

Cleared Events

Action	URI	Method
View/search/filter the list of cleared events.	/cleared_event	GET
View the properties of a cleared event.	/cleared_event/X	GET

Collection Labels

Action	URI	Method
View/search/filter the list of collection labels.	/collection_label	GET
View the properties of a collection label.	/collection_label/X	GET

Collection Label Groups

Action	URI	Method
View/search/filter the list of collection label groups.	/collection_label_group	GET
View the properties of a collection label group.	/collection_label_group/X	GET

Collector Groups

Action	URI	Method
View/search/filter the list of collector groups.	/collector_group	GET
Create a new collector group.	/collector_group	POST
View the properties of a collector group.	/collector_group/X	GET
Update the properties of a collector group.	/collector_group/X	POST
Replace a collector group.	/collector_group/X	PUT
Delete a collector group.	/collector_group/X	DELETE

To enable multi-tenancy for collector groups, the database setting "master.system_settings_core.enable_cug_orgs" must be set to 1. When multi-tenancy is enabled, an administrative user can update all collector groups using the new fields. Non-administrative users can update all collector groups for which the "all_orgs" field is set to 1. Otherwise, these users can only update credentials and collector groups within their aligned organizations. Be aware that you might encounter a situation where a device is not aligned to a collector group if you do not properly configure these actions.		
View the properties of all collector groups.	/collector_group/all_orgs	GET
Update the properties of all collector groups.	/collector_group/all_orgs	POST
View the properties of only the collector groups within your aligned organizations.	/collector_group/aligned_organizations	GET
Update the properties of only the collector groups within your aligned organizations	/collector_group/aligned_organizations	POST

Credentials

Action	URI	Method
View the index of available credential resources.	/credential	GET
View/search/filter the list of basic/snippet credentials.	/credential/basic	GET
Create a new basic/snippet credential.	/credential/basic	POST
View a basic/snippet credential.	/credential/basic/X	GET
Update a basic/snippet credential.	/credential/basic/X	POST
Replace a basic/snippet credential.	/credential/basic/X	PUT
Delete a basic/snippet credential.	/credential/basic/X	DELETE
View/search/filter the list of database credentials.	/credential/db	GET
Create a new database credential.	/credential/db	POST
View a database credential.	/credential/db/X	GET

Action	URI	Method
Update a database credential.	/credential/db/X	POST
Replace a database credential.	/credential/db/X	PUT
Delete a database credential.	/credential/db/X	DELETE
View/search/filter the list of LDAP/AD credentials.	/credential/ldap	GET
Create a new LDAP/AD credential.	/credential/ldap	POST
View a LDAP/AD credential.	/credential/ldap/X	GET
Update a LDAP/AD credential.	/credential/ldap/X	POST
Replace a LDAP/AD credential.	/credential/ldap/X	PUT
Delete a LDAP/AD credential.	/credential/ldap/X	DELETE
View/search/filter the list of PowerShell credentials.	/credential/powershell	GET
Create a new PowerShell credential.	/credential/powershell	POST
View a PowerShell credential.	/credential/powershell/X	GET
Update a PowerShell credential.	/credential/powershell/X	POST
Replace a PowerShell credential.	/credential/powershell/X	PUT
Delete a PowerShell credential.	/credential/powershell/X	DELETE
View/search/filter the list of SNMP credentials.	/credential/snmp	GET
Create a new SNMP credential.	/credential/snmp	POST
View an SNMP credential.	/credential/snmp/X	GET
Update an SNMP credential.	/credential/snmp/X	POST
Replace an SNMP credential.	/credential/snmp/X	PUT
Delete an SNMP credential.	/credential/snmp/X	DELETE
View/search/filter the list of SOAP/XML credentials.	/credential/soap	GET
Create a new SOAP/XML credential.	/credential/soap	POST
View a SOAP/XML credential.	/credential/soap/X	GET
Update a SOAP/XML credential.	/credential/soap/X	POST
Replace a SOAP/XML credential.	/credential/soap/X	PUT

Action	URI	Method
Delete a SOAP/XML credential.	/credential/soap/X	DELETE
View/search/filter the list of SSH credentials.	/credential/ssh	GET
Create a new SSH credential.	/credential/ssh	POST
View an SSH credential.	/credential/ssh/X	GET
Update an SSH credential.	/credential/ssh/X	POST
Replace an SSH credential.	/credential/ssh/X	PUT
Delete an SSH credential.	/credential/ssh/X	DELETE

Custom Attributes

Action	URI	Method
View the index of available custom attribute resources.	/custom_attribute	GET
View the custom attributes defined for assets.	/custom_attribute/asset	GET
Add a custom attribute for assets.	/custom_attribute/asset	POST
View details of a custom attribute defined for assets.	/custom_attribute/asset/X	GET
Update a custom attribute defined for assets.	/custom_attribute/asset/X	POST
Delete a custom attribute defined for assets.	/custom_attribute/asset/X	DELETE
View example JSON or XML content for creating custom attributes for assets.	/custom_attribute/asset/_example	GET
View the custom attributes defined for devices.	/custom_attribute/device	GET
Add a custom attribute for devices.	/custom_attribute/device	POST
View details of a custom attribute defined for devices.	/custom_attribute/device/X	GET

Action	URI	Method
Update a custom attribute defined for devices.	/custom_attribute/device/X	POST
Delete a custom attribute defined for devices.	/custom_attribute/device/X	DELETE
View example JSON or XML content for creating custom attributes for devices.	/custom_attribute/device/_example	GET
View the custom attributes defined for interfaces.	/custom_attribute/interface	GET
Add a custom attribute for interfaces.	/custom_attribute/interface	POST
View details of a custom attribute defined for interfaces.	/custom_attribute/interface/X	GET
Update a custom attribute defined for interfaces.	/custom_attribute/interface/X	POST
Delete a custom attribute defined for interfaces.	/custom_attribute/interface/X	DELETE
View example JSON or XML content for creating custom attributes for interfaces.	/custom_attribute/interface/_example	GET
View the custom attributes defined for themes.	/custom_attribute/theme	GET
Add a custom attribute for themes.	/custom_attribute/theme	POST
View details of a custom attribute defined for themes.	/custom_attribute/theme/X	GET
Update a custom attribute defined for themes.	/custom_attribute/theme/X	POST
Delete a custom attribute defined for themes.	/custom_attribute/theme/X	DELETE
View example JSON or XML content for creating custom attributes for themes.	/custom_attribute/theme/_example	GET
View the custom attributes defined for vendors.	/custom_attribute/vendor	GET
Add a custom attribute for vendors.	/custom_attribute/vendor	POST
View details of a custom attribute defined for vendors.	/custom_attribute/vendor/X	GET

Action	URI	Method
Update a custom attribute defined for vendors.	/custom_attribute/vendor/X	POST
Delete a custom attribute defined for vendors.	/custom_attribute/vendor/X	DELETE
View example JSON or XML content for creating custom attributes for vendors.	/custom_attribute/vendor/_example	GET
View the custom attributes defined for all entity types.	/custom_attribute/_lookup	GET

Dashboards

Action	URI	Method
View/search/filter the list of dashboards.	/dashboard	GET
Create a new dashboard.	/dashboard	POST
View the properties of a dashboard.	/dashboard/X	GET
Update the properties of a dashboard.	/dashboard/X	POST
Replace a dashboard.	/dashboard/X	PUT
Delete a dashboard.	/dashboard/X	DELETE
View/search/filter the list of widgets on a dashboard.	/dashboard/X/widget	GET
View the properties of a widget on a dashboard.	/dashboard/X/widget/X	GET
Update the properties of a widget on a dashboard.	/dashboard/X/widget/X	POST
Replace a widget on a dashboard.	/dashboard/X/widget/X	PUT
Remove a widget from a dashboard.	/dashboard/X/widget/X	DELETE
Create a new dashboard by duplicating an existing dashboard.	/dashboard	POST a /dashboard resource.

Devices

Action	URI	Method
View/search/filter the list of devices.	/device	GET
Create a new virtual device.	/device	POST
View the properties of a device.	/device/X	GET
Update the properties of a device.	/device/X	POST
Replace the properties of a device.	/device/X	PUT
Delete a device.	/device/X	DELETE
View/search/filter the list of Dynamic Applications aligned with a device.	/device/X/aligned_app	GET
Align a Dynamic Application with a device.	/device/X/aligned_app	POST
View the collection status and associated credential for a Dynamic Application aligned with a device.	/device/X/aligned_app/X	GET
Update the collection status and associated credential for a Dynamic Application aligned with a device.	/device/X/aligned_app/X	POST
Unalign a Dynamic Application from a device.	/device/X/aligned_app/X	DELETE
View/search/filter the list of available configuration data for a device.	/device/X/config_data	GET
View meta-data about data collected from a device by a configuration Dynamic Application.	/device/X/config_data/X	GET
View data collected from a device by a configuration Dynamic Application.	/device/X/config_data/X/data	GET
View historical snapshots of data collected from a device by a configuration Dynamic Application.	/device/X/config_data/X/snapshots	GET
View general information collected from a device.	/device/X/detail	GET

Action	URI	Method
View/search/filter the list of credentials aligned with a device.	/device/X/device_app_credentials	GET
View the threshold settings for a device.	/device/X/device_thresholds	GET
Update the threshold settings for a device.	/device/X/device_thresholds	POST
Replace the threshold settings for a device.	/device/X/device_thresholds	PUT
Revert all device thresholds to the global default values.	/device/X/device_thresholds	DELETE
Add an interface record to a device.	/device/X/interface	POST
View/search/filter the list of interfaces for a device.	/device/X/interface	GET
View the properties of an interface for a device, including all interface tags.	/device/X/interface/X	GET
Update the properties of an interface for a device. This can create a new interface without an interface tag or create a new interface by referencing an existing interface tag.	/device/X/interface/X	POST
Replace an interface record associated with a device. This can update an interface without affecting the interface tag association.	/device/X/interface/X	PUT
Delete an interface record associated with a device. Deleting an interface also deletes the interface tag.	/device/X/interface/X	DELETE
View data for an interface.	/device/X/interface/X/interface_data/data	GET
View daily normalized data for an interface.	/device/X/interface/X/interface_data/normalized_daily	GET
View hourly normalized data for an interface.	/device/X/interface/X/interface_data/normalized_hourly	GET
View/search/filter the list of logs associated with a device.	/device/X/log/	GET
View a log associated with a device.	/device/X/log/X	GET
Add a note to a device.	/device/X/note/	POST

Action	URI	Method
View/search/filter the list of notes associated with a device.	/device/X/note/	GET
View a note associated with a device.	/device/X/note/X	GET
Update a note associated with a device.	/device/X/note/X	POST
Replace a note associated with a device.	/device/X/note/X	PUT
View/search/filter the list of files associated with a device note.	/device/X/note/X/media	GET
Get a media file associated with a device note.	/device/X/note/X/media/X	GET
Add a media file to a device note.	/device/X/note/X/media/X	PUT
View meta-data about a media file associated with a device note.	/device/X/note/X/media/X/info	GET
View/search/filter the list of available Dynamic Application data for a device.	/device/X/performance_data	GET
View data for a Dynamic Application aligned to a device.	/device/X/performance_data/X/data	GET
View daily normalized data for a Dynamic Application aligned to a device.	/device/X/performance_data/X/normalized_daily	GET
View hourly normalized data for a Dynamic Application aligned to a device.	/device/X/performance_data/X/normalized_hourly	GET
View/search/filter the list of available vitals data for a device.	/device/X/vitals	GET
View availability data for a device.	/device/X/vitals/availability/data	GET
View daily normalized availability data for a device.	/device/X/vitals/availability/normalized_daily	GET
View hourly normalized availability data for a device.	/device/X/vitals/availability/normalized_hourly	GET
View data for a file system on a device.	/device/X/vitals/fsX/data	GET
View daily normalized data for a file system on a device.	/device/X/vitals/fsX/normalized_daily	GET
View latency data for a device.	/device/X/vitals/latency/data	GET

Action	URI	Method
View daily normalized latency data for a device.	/device/X/vitals/latency/normalized_daily	GET
View hourly normalized latency data for a device.	/device/X/vitals/latency/normalized_hourly	GET
Apply a device template to a device.	/device/X	Post a /device_template resource.

Device Categories

Action	URI	Method
View/search/filter the list of device categories.	/device_category	GET
View the properties of a device category.	/device_category/X	GET

Device Classes

Action	URI	Method
View/search/filter the list of device classes.	/device_class	GET
View the properties of a device class.	/device_class/X	GET

Device Groups

Action	URI	Method
View/search/filter the list of device groups.	/device_group	GET
Create a new device group.	/device_group	POST

Action	URI	Method
View the properties of a device group.	/device_group/X	GET
Update the properties of a device group.	/device_group/X	POST
Replace a device group.	/device_group/X	PUT
Delete a device group.	/device_group/X	DELETE
View a list of all devices in the device group, including devices that match dynamic rules.	/device_group/X/expanded_devices	GET
Apply a device template to a device group.	/device_group/X	Post a /device_template resource.

Device Relationships

Action	URI	Method
View/search/filter the list of device relationships.	/relationship	GET
View the properties of a device relationship.	/relationship/X	GET
View/search/filter the list of ancestor and descendant devices of a device.	/relationship_hierarchy/X	GET

Device Relationship Types

Action	URI	Method
View/search/filter the list of device relationship types.	/relationship_type	GET
View the properties of a device relationship type.	/relationship_type/X	GET

Device Templates

Action	URI	Method
View/search/filter the list of device templates.	/device_template	GET
Create a new device template.	/device_template	POST
View the properties of a device template.	/device_template/X	GET
Update the properties of a device template.	/device_template/X	POST
Replace a device template.	/device_template/X	PUT
Delete a device template.	/device_template/X	DELETE
View/search/filter the list of web content monitoring policy sub-templates associated with a device template.	/device_template/X/subtpl_cv	GET
Create a new web content monitoring policy sub-template for a device template.	/device_template/X/subtpl_cv	POST
View the properties of a web content monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_cv/X	GET
Update a web content monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_cv/X	POST
Replace a web content monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_cv/X	PUT
Delete a web content monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_cv/X	DELETE
View/search/filter the list of Dynamic Application sub-templates associated with a device template.	/device_template/X/subtpl_dynapp	GET
Create a new Dynamic Application sub-template for a device template.	/device_template/X/subtpl_dynapp	POST

Action	URI	Method
View the properties of a Dynamic Application sub-template associated with a device template.	/device_template/X/subtpl_dynapp/X	GET
Update a Dynamic Application sub-template associated with a device template.	/device_template/X/subtpl_dynapp/X	POST
Replace a Dynamic Application sub-template associated with a device template.	/device_template/X/subtpl_dynapp/X	PUT
Delete a Dynamic Application sub-template associated with a device template.	/device_template/X/subtpl_dynapp/X	DELETE
View/search/filter the list of port monitoring policy sub-templates associated with a device template.	/device_template/X/subtpl_port	GET
Create a new port monitoring policy sub-template for a device template.	/device_template/X/subtpl_port	POST
View the properties of a port monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_port/X	GET
Update a port monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_port/X	POST
Replace a port monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_port/X	PUT
Delete a port monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_port/X	DELETE
View/search/filter the list of process monitoring policy sub-templates associated with a device template.	/device_template/X/subtpl_process	GET
Create a new process monitoring policy sub-template for a device template.	/device_template/X/subtpl_process	POST
View the properties of a process monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_process/X	GET

Action	URI	Method
Update a process monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_process/X	POST
Replace a process monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_process/X	PUT
Delete a process monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_process/X	DELETE
View/search/filter the list of Windows service monitoring policy sub-templates associated with a device template.	/device_template/X/subtpl_service	GET
Create a new Windows service monitoring policy sub-template for a device template.	/device_template/X/subtpl_service	POST
View the properties of a Windows service monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_service/X	GET
Update a Windows service monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_service/X	POST
Replace a Windows service monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_service/X	PUT
Delete a Windows service monitoring policy sub-template associated with a device template.	/device_template/X/subtpl_service/X	DELETE

Discovery Sessions

Action	URI	Method
View/search/filter the list of discovery sessions.	/discovery_session	GET
Create a new discovery session.	/discovery_session	POST

Action	URI	Method
View the properties of a discovery session.	/discovery_session/X	GET
Update a discovery session.	/discovery_session/X	POST
Replace a discovery session.	/discovery_session/X	PUT
Delete a discovery session.	/discovery_session/X	DELETE
View/search/filter the list of logs associated with a discovery session.	/discovery_session/X/log	GET
View a log message associated with a discovery session.	/discovery_session/X/log/X	GET
View/search/filter the list of currently running discovery sessions.	/discovery_session_active	GET
Create and immediately run a new discovery session.	/discovery_session_active	POST
View the properties of a currently running discovery session.	/discovery_session_active/X	GET
Stop a currently running discovery session.	/discovery_session_active/X	DELETE
View/search/filter the list of logs associated with a currently running discovery session.	/discovery_session_active/X/log	GET
View a log message associated with a currently running discovery session.	/discovery_session_active/X/log/X	GET
Start a discovery session.	/discovery_session_active	POST a /discovery_session resource.

Dynamic Applications

Action	URI	Method
View the index of available Dynamic Application resources.	/dynamic_app	GET

Action	URI	Method
View/search/filter the list of Database Configuration Dynamic Applications.	/dynamic_app/db_config	GET
View the properties of a Database Configuration Dynamic Application.	/dynamic_app/db_config/X	GET
View/search/filter the list of collection objects associated with a Database Configuration Dynamic Application.	/dynamic_app/db_config/X/collection_object	GET
Add a collection object to a Database Configuration Dynamic Application.	/dynamic_app/db_config/X/collection_object	POST
View the properties of a collection object associated with a Database Configuration Dynamic Application.	/dynamic_app/db_config/X/collection_object/X	GET
Update the properties of a collection object associated with a Database Configuration Dynamic Application.	/dynamic_app/db_config/X/collection_object/X	POST
Replace a collection object associated with a Database Configuration Dynamic Application.	/dynamic_app/db_config/X/collection_object/X	PUT
Remove a collection object from a Database Configuration Dynamic Application.	/dynamic_app/db_config/X/collection_object/X	DELETE
View/search/filter the list of Database Performance Dynamic Applications.	/dynamic_app/db_performance	GET
View the properties of a Database Performance Dynamic Application.	/dynamic_app/db_performance/X	GET
View/search/filter the list of collection objects associated with a Database Performance Dynamic Application.	/dynamic_app/db_performance/X/collection_object	GET
Add a collection object to a Database Performance Dynamic Application.	/dynamic_app/db_performance/X/collection_object	POST
View the properties of a collection object associated with a Database Performance Dynamic Application.	/dynamic_app/db_performance/X/collection_object/X	GET
Update the properties of a collection object associated with a Database Performance Dynamic Application.	/dynamic_app/db_performance/X/collection_object/X	POST

Action	URI	Method
Replace a collection object associated with a Database Performance Dynamic Application.	/dynamic_app/db_performance/X/collection_object/X	PUT
Remove a collection object from a Database Performance Dynamic Application.	/dynamic_app/db_performance/X/collection_object/X	DELETE
View/search/filter the list of presentation objects associated with a Database Performance Dynamic Application.	/dynamic_app/db_performance/X/presentation_object	GET
Add a presentation object to a Database Performance Dynamic Application.	/dynamic_app/db_performance/X/presentation_object	POST
View the properties of a presentation object associated with a Database Performance Dynamic Application.	/dynamic_app/db_performance/X/presentation_object/X	GET
Update the properties of a presentation object associated with a Database Performance Dynamic Application.	/dynamic_app/db_performance/X/presentation_object/X	POST
Replace a presentation object associated with a Database Performance Dynamic Application.	/dynamic_app/db_performance/X/presentation_object/X	PUT
Remove a presentation object from a Database Performance Dynamic Application.	/dynamic_app/db_performance/X/presentation_object/X	DELETE
View/search/filter the list of PowerShell Configuration Dynamic Applications.	/dynamic_app/powershell_config	GET
View the properties of a PowerShell Configuration Dynamic Application.	/dynamic_app/powershell_config/X	GET
View/search/filter the list of collection objects associated with a PowerShell Configuration Dynamic Application.	/dynamic_app/powershell_config/X/collection_object	GET
Add a collection object to a PowerShell Configuration Dynamic Application.	/dynamic_app/powershell_config/X/collection_object	POST
View the properties of a collection object associated with a PowerShell Configuration Dynamic Application.	/dynamic_app/powershell_config/X/collection_object/X	GET

Action	URI	Method
Update the properties of a collection object associated with a PowerShell Configuration Dynamic Application.	/dynamic_app/powershell_config/X/collection_object/X	POST
Replace a collection object associated with a PowerShell Configuration Dynamic Application.	/dynamic_app/powershell_config/X/collection_object/X	PUT
Remove a collection object from a PowerShell Configuration Dynamic Application.	/dynamic_app/powershell_config/X/collection_object/X	DELETE
View/search/filter the list of PowerShell Performance Dynamic Applications.	/dynamic_app/powershell_performance	GET
View the properties of a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X	GET
View/search/filter the list of collection objects associated with a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X/collection_object	GET
Add a collection object to a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X/collection_object	POST
View the properties of a collection object associated with a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X/collection_object/X	GET
Update the properties of a collection object associated with a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X/collection_object/X	POST
Replace a collection object associated with a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X/collection_object/X	PUT
Remove a collection object from a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X/collection_object/X	DELETE
View/search/filter the list of presentation objects associated with a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X/presentation_object	GET
Add a presentation object to a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X/presentation_object	POST

Action	URI	Method
View the properties of a presentation object associated with a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X/presentation_object/X	GET
Update the properties of a presentation object associated with a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X/presentation_object/X	POST
Replace a presentation object associated with a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X/presentation_object/X	PUT
Remove a presentation object from a PowerShell Performance Dynamic Application.	/dynamic_app/powershell_performance/X/presentation_object/X	DELETE
View/search/filter the list of Snippet Configuration Dynamic Applications.	/dynamic_app/snippet_config	GET
View the properties of a Snippet Configuration Dynamic Application.	/dynamic_app/snippet_config/X	GET
View/search/filter the list of collection objects associated with a Snippet Configuration Dynamic Application.	/dynamic_app/snippet_config/X/collection_object	GET
Add a collection object to a Snippet Configuration Dynamic Application.	/dynamic_app/snippet_config/X/collection_object	POST
View the properties of a collection object associated with a Snippet Configuration Dynamic Application.	/dynamic_app/snippet_config/X/collection_object/X	GET
Update the properties of a collection object associated with a Snippet Configuration Dynamic Application.	/dynamic_app/snippet_config/X/collection_object/X	POST
Replace a collection object associated with a Snippet Configuration Dynamic Application.	/dynamic_app/snippet_config/X/collection_object/X	PUT
Remove a collection object from a Snippet Configuration Dynamic Application.	/dynamic_app/snippet_config/X/collection_object/X	DELETE
View/search/filter the list of Snippet Journal Dynamic Applications.	/dynamic_app/snippet_journal	GET
View the properties of a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X	GET

Action	URI	Method
View/search/filter the list of collection objects associated with a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X/collection_object	GET
Add a collection object to a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X/collection_object	POST
View the properties of a collection object associated with a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X/collection_object/X	GET
Update the properties of a collection object associated with a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X/collection_object/X	POST
Replace a collection object associated with a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X/collection_object/X	PUT
Remove a collection object from a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X/collection_object/X	DELETE
Add a presentation object to a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X/presentation_object	POST
View/search/filter the list of presentation objects associated with a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X/presentation_object	GET
View the properties of a presentation object associated with a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X/presentation_object/X	GET
Update the properties of a presentation object associated with a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X/presentation_object/X	POST
Replace a presentation object associated with a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X/presentation_object/X	PUT
Remove a presentation object from a Snippet Journal Dynamic Application.	/dynamic_app/snippet_journal/X/presentation_object/X	DELETE
View/search/filter the list of Snippet Performance Dynamic Applications.	/dynamic_app/snippet_performance	GET
View the properties of a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X	GET

Action	URI	Method
View/search/filter the list of collection objects associated with a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X/collection_object	GET
Add a collection object to a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X/collection_object	POST
View the properties of a collection object associated with a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X/collection_object/X	GET
Update the properties of a collection object associated with a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X/collection_object/X	POST
Replace a collection object associated with a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X/collection_object/X	PUT
Remove a collection object from a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X/collection_object/X	DELETE
View/search/filter the list of presentation objects associated with a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X/presentation_object	GET
Add a presentation object to a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X/presentation_object	POST
View the properties of a presentation object associated with a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X/presentation_object/X	GET
Update the properties of a presentation object associated with a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X/presentation_object/X	POST
Replace a presentation object associated with a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X/presentation_object/X	PUT
Remove a presentation object from a Snippet Performance Dynamic Application.	/dynamic_app/snippet_performance/X/presentation_object/X	DELETE
View/search/filter the list of SNMP Configuration Dynamic Applications.	/dynamic_app/snmp_config	GET

Action	URI	Method
View the properties of an SNMP Configuration Dynamic Application.	/dynamic_app/snmp_config/X	GET
View/search/filter the list of collection objects associated with an SNMP Configuration Dynamic Application.	/dynamic_app/snmp_config/X/collection_object	GET
Add a collection object to an SNMP Configuration Dynamic Application.	/dynamic_app/snmp_config/X/collection_object	POST
View the properties of a collection object associated with an SNMP Configuration Dynamic Application.	/dynamic_app/snmp_config/X/collection_object/X	GET
Update the properties of a collection object associated with an SNMP Configuration Dynamic Application.	/dynamic_app/snmp_config/X/collection_object/X	POST
Replace a collection object associated with an SNMP Configuration Dynamic Application.	/dynamic_app/snmp_config/X/collection_object/X	PUT
Remove a collection object from an SNMP Configuration Dynamic Application.	/dynamic_app/snmp_config/X/collection_object/X	DELETE
View/search/filter the list of SNMP Performance Dynamic Applications.	/dynamic_app/snmp_performance	GET
View the properties of an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X	GET
View/search/filter the list of collection objects associated with an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X/collection_object	GET
Add a collection object to an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X/collection_object	POST
View the properties of a collection object associated with an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X/collection_object/X	GET
Update the properties of a collection object associated with an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X/collection_object/X	POST
Replace a collection object associated with an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X/collection_object/X	PUT

Action	URI	Method
Remove a collection object from an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X/collection_object/X	DELETE
View/search/filter the list of presentation objects associated with an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X/presentation_object	GET
Add a presentation object to an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X/presentation_object	POST
View the properties of a presentation object associated with an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X/presentation_object/X	GET
Update the properties of a presentation object associated with an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X/presentation_object/X	POST
Replace a presentation object associated with an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X/presentation_object/X	PUT
Remove a presentation object from an SNMP Performance Dynamic Application.	/dynamic_app/snmp_performance/X/presentation_object/X	DELETE
View/search/filter the list of SOAP Configuration Dynamic Applications.	/dynamic_app/soap_config	GET
View the properties of a SOAP Configuration Dynamic Application.	/dynamic_app/soap_config/X	GET
Add a collection object to a SOAP Configuration Dynamic Application.	/dynamic_app/soap_config/X/collection_object	POST
View/search/filter the list of collection objects associated with a SOAP Configuration Dynamic Application.	/dynamic_app/soap_config/X/collection_object	GET
View the properties of a collection object associated with a SOAP Configuration Dynamic Application.	/dynamic_app/soap_config/X/collection_object/X	GET
Update the properties of a collection object associated with a SOAP Configuration Dynamic Application.	/dynamic_app/soap_config/X/collection_object/X	POST

Action	URI	Method
Replace a collection object associated with a SOAP Configuration Dynamic Application.	/dynamic_app/soap_config/X/collection_object/X	PUT
Remove a collection object from a SOAP Configuration Dynamic Application.	/dynamic_app/soap_config/X/collection_object/X	DELETE
View/search/filter the list of SOAP Performance Dynamic Applications.	/dynamic_app/soap_performance	GET
View the properties of a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X	GET
View/search/filter the list of collection objects associated with a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X/collection_object	GET
Add a collection object to a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X/collection_object	POST
View the properties of a collection object associated with a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X/collection_object/X	GET
Update the properties of a collection object associated with a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X/collection_object/X	POST
Replace a collection object associated with a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X/collection_object/X	PUT
Remove a collection object from a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X/collection_object/X	DELETE
View/search/filter the list of presentation objects associated with a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X/presentation_object	GET
Add a presentation object to a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X/presentation_object	POST
View the properties of a presentation object associated with a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X/presentation_object/X	GET
Update the properties of a presentation object associated with a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X/presentation_object/X	POST

Action	URI	Method
Replace a presentation object associated with a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X/presentation_object/X	PUT
Remove a presentation object from a SOAP Performance Dynamic Application.	/dynamic_app/soap_performance/X/presentation_object/X	DELETE
View/search/filter the list of WMI Configuration Dynamic Applications.	/dynamic_app/wmi_config	GET
View the properties of a WMI Configuration Dynamic Application.	/dynamic_app/wmi_config/X	GET
View/search/filter the list of collection objects associated with a WMI Configuration Dynamic Application.	/dynamic_app/wmi_config/X/collection_object	GET
Add a collection object to a WMI Configuration Dynamic Application.	/dynamic_app/wmi_config/X/collection_object	POST
View the properties of a collection object associated with a WMI Configuration Dynamic Application.	/dynamic_app/wmi_config/X/collection_object/X	GET
Update the properties of a collection object associated with a WMI Configuration Dynamic Application.	/dynamic_app/wmi_config/X/collection_object/X	POST
Replace a collection object associated with a WMI Configuration Dynamic Application.	/dynamic_app/wmi_config/X/collection_object/X	PUT
Remove a collection object from a WMI Configuration Dynamic Application.	/dynamic_app/wmi_config/X/collection_object/X	DELETE
View/search/filter the list of WMI Performance Dynamic Applications.	/dynamic_app/wmi_performance	GET
View the properties of a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X	GET
View/search/filter the list of collection objects associated with a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X/collection_object	GET
Add a collection object to a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X/collection_object	POST

Action	URI	Method
View the properties of a collection object associated with a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X/collection_object/X	GET
Update the properties of a collection object associated with a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X/collection_object/X	POST
Replace a collection object associated with a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X/collection_object/X	PUT
Remove a collection object from a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X/collection_object/X	DELETE
View/search/filter the list of presentation objects associated with a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X/presentation_object	GET
Add a presentation object to a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X/presentation_object	POST
View the properties of a presentation object associated with a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X/presentation_object/X	GET
Update the properties of a presentation object associated with a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X/presentation_object/X	POST
Replace a presentation object associated with a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X/presentation_object/X	PUT
Remove a presentation object from a WMI Performance Dynamic Application.	/dynamic_app/wmi_performance/X/presentation_object/X	DELETE
View/search/filter the list of XML Configuration Dynamic Applications.	/dynamic_app/xml_config	GET
View the properties of an XML Configuration Dynamic Application.	/dynamic_app/xml_config/X	GET
Add a collection object to an XML Configuration Dynamic Application.	/dynamic_app/xml_config/X/collection_object	POST
View/search/filter the list of collection objects associated with an XML Configuration Dynamic Application.	/dynamic_app/xml_config/X/collection_object	GET

Action	URI	Method
View the properties of a collection object associated with an XML Configuration Dynamic Application.	/dynamic_app/xml_config/X/collection_object/X	GET
Update the properties of a collection object associated with an XML Configuration Dynamic Application.	/dynamic_app/xml_config/X/collection_object/X	POST
Replace a collection object associated with an XML Configuration Dynamic Application.	/dynamic_app/xml_config/X/collection_object/X	PUT
Remove a collection object from an XML Configuration Dynamic Application.	/dynamic_app/xml_config/X/collection_object/X	DELETE
View/search/filter the list of XML Performance Dynamic Applications.	/dynamic_app/xml_performance	GET
View the properties of an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X	GET
View/search/filter the list of collection objects associated with an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X/collection_object	GET
Add a collection object to an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X/collection_object	POST
View the properties of a collection object associated with an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X/collection_object/X	GET
Update the properties of a collection object associated with an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X/collection_object/X	POST
Replace a collection object associated with an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X/collection_object/X	PUT
Remove a collection object from an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X/collection_object/X	DELETE
View/search/filter the list of presentation objects associated with an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X/presentation_object	GET
Add a presentation object to an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X/presentation_object	POST

Action	URI	Method
View the properties of a presentation object associated with an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X/presentation_object/X	GET
Update the properties of a presentation object associated with an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X/presentation_object/X	POST
Replace a presentation object associated with an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X/presentation_object/X	PUT
Remove a presentation object from an XML Performance Dynamic Application.	/dynamic_app/xml_performance/X/presentation_object/X	DELETE
View/search/filter the list of XSLT Configuration Dynamic Applications.	/dynamic_app/xslt_config	GET
View the properties of an XSLT Configuration Dynamic Application.	/dynamic_app/xslt_config/X	GET
View/search/filter the list of collection objects associated with an XSLT Configuration Dynamic Application.	/dynamic_app/xslt_config/X/collection_object	GET
Add a collection object to an XSLT Configuration Dynamic Application.	/dynamic_app/xslt_config/X/collection_object	POST
View the properties of a collection object associated with an XSLT Configuration Dynamic Application.	/dynamic_app/xslt_config/X/collection_object/X	GET
Update the properties of a collection object associated with an XSLT Configuration Dynamic Application.	/dynamic_app/xslt_config/X/collection_object/X	POST
Replace a collection object associated with a Dynamic Application.	/dynamic_app/xslt_config/X/collection_object/X	PUT
Remove a collection object from an XSLT Configuration Dynamic Application.	/dynamic_app/xslt_config/X/collection_object/X	DELETE
View/search/filter the list of XSLT Performance Dynamic Applications.	/dynamic_app/xslt_performance	GET
View the properties of an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X	GET

Action	URI	Method
View/search/filter the list of collection objects associated with an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X/collection_object	GET
Add a collection object to an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X/collection_object	POST
View the properties of a collection object associated with an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X/collection_object/X	GET
Update the properties of a collection object associated with an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X/collection_object/X	POST
Replace a collection object associated with an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X/collection_object/X	PUT
Remove a collection object from an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X/collection_object/X	DELETE
View/search/filter the list of presentation objects associated with an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X/presentation_object	GET
Add a presentation object to an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X/presentation_object	POST
View the properties of a presentation object associated with an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X/presentation_object/X	GET
Update the properties of a presentation object associated with an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X/presentation_object/X	POST
Replace a presentation object associated with an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X/presentation_object/X	PUT
Remove a presentation object from an XSLT Performance Dynamic Application.	/dynamic_app/xslt_performance/X/presentation_object/X	DELETE
View/search/filter the list of all Dynamic Applications.	/dynamic_app/_lookup	GET

Events

Action	URI	Method
View/search/filter the list of active events.	/event	GET
View an active event.	/event/X	GET
Clear an active event.	/event/X	DELETE
Update the properties of an event.	/event/X	POST

Event Categories

Action	URI	Method
Add an event category to an active event.	/event_category/X	POST
View the event category for an active event.	/event_category/X	GET
Delete the event category for an active event.	/event_category/X	DELETE
Update the properties of an event.	/event_category/X	POST

External Contacts

Action	URI	Method
View/search/filter the list of external contacts.	/contacts	GET
Create a new external contact.	/contacts	POST
View the properties of an external contact.	/contacts/X	GET

Action	URI	Method
Update the properties of an external contact.	/contacts/X	POST
Replace an external contact.	/contacts/X	PUT
Delete an external contact.	/contacts/X	DELETE

File Uploads

Action	URI	Method
View the index of available filestore resources.	/filestore	GET
View the index of available PowerPack file resources. This index does not include PowerPacks that are automatically installed by ScienceLogic patches.	/filestore/powerpack	GET
Download a PowerPack file.	/filestore/powerpack/X	GET
View the information associated with a PowerPack file.	/filestore/powerpack/X/info	GET
View the index of available patch file resources.	/filestore/system_patch	GET
Download a patch file.	/filestore/system_patch/X	GET
View the information associated with a patch file.	/filestore/system_patch/X/info	GET

Interfaces

Action	URI	Method
View/search/filter the list of interfaces.	/interface	GET
Add an interface record to a device.	/interface	POST

Action	URI	Method
View the properties of an interface.	/interface/X	GET
Update the properties of an interface.	/interface/X	POST
Replace an interface record.	/interface/X	PUT
Delete an interface record.	/interface/X	DELETE
View data for an interface.	/interface/X/interface_data/data	GET
View daily normalized data for an interface.	/interface/X/interface_data/normalized_daily	GET
View hourly normalized data for an interface.	/interface/X/interface_data/normalized_hourly	GET

Interface Metrics

Action	URI	Method
View/search/filter the list of interface metrics.	/interface_metric	GET
View details about an interface metric.	/interface_metric/X	GET

Interface Tags

Action	URI	Method
View/search/filter the list of interface tags and their names.	/interface_tag	GET
Add a new interface tag.	/interface_tag	POST
Update the name of an interface tag.	/interface_tag	PUT
Delete an interface tag. You cannot delete a tag mapped to an interface.	/interface_tag	DELETE

Monitors

Action	URI	Method
View the index of available monitoring policy resources.	/monitor	GET
View/search/filter the list of web content monitoring policies.	/monitor/cv	GET
Create a new web content monitoring policy.	/monitor/cv	POST
View a web content monitoring policy.	/monitor/cv/X	GET
Update a web content monitoring policy.	/monitor/cv/X	POST
Replace a web content monitoring policy.	/monitor/cv/X	PUT
Delete a web content monitoring policy.	/monitor/cv/X	DELETE
View/search/filter the list of domain name monitoring policies.	/monitor/dns	GET
Create a new domain name monitoring policy.	/monitor/dns	POST
View a domain name monitoring policy.	/monitor/dns/X	GET
Update a domain name monitoring policy.	/monitor/dns/X	POST
Replace a domain name monitoring policy.	/monitor/dns/X	PUT
Delete a domain name monitoring policy.	/monitor/dns/X	DELETE
View/search/filter the list of Email round-trip monitoring policies.	/monitor/email	GET
Create a new Email round-trip monitoring policy.	/monitor/email	POST
View an Email round-trip monitoring policy.	/monitor/email/X	GET
Update an Email round-trip monitoring policy.	/monitor/email/X	POST

Action	URI	Method
Replace an Email round-trip monitoring policy.	/monitor/email/X	PUT
Delete an Email round-trip monitoring policy.	/monitor/email/X	DELETE
View/search/filter the list of port monitoring policies.	/monitor/port	GET
Create a new port monitoring policy.	/monitor/port	POST
View a port monitoring policy.	/monitor/port/X	GET
Update a port monitoring policy.	/monitor/port/X	POST
Replace a port monitoring policy.	/monitor/port/X	PUT
Delete a port monitoring policy.	/monitor/port/X	DELETE
Create a new system process monitoring policy.	/monitor/process	POST
View/search/filter the list of system process monitoring policies.	/monitor/process	GET
View a system process monitoring policy.	/monitor/process/X	GET
Update a system process monitoring policy.	/monitor/process/X	POST
Replace a system process monitoring policy.	/monitor/process/X	PUT
Delete a system process monitoring policy.	/monitor/process/X	DELETE
View/search/filter the list of Windows service monitoring policies.	/monitor/service	GET
Create a new Windows service monitoring policy.	/monitor/service	POST
View a Windows service monitoring policy.	/monitor/service/X	GET
Update a Windows service monitoring policy.	/monitor/service/X	POST
Replace a Windows service monitoring policy.	/monitor/service/X	PUT

Action	URI	Method
Delete a Windows service monitoring policy.	/monitor/service/X	DELETE
View/search/filter the list of SOAP/XML transaction monitoring policies.	/monitor/tv	GET
Create a new SOAP/XML transaction monitoring policy.	/monitor/tv	POST
View a SOAP/XML transaction monitoring policy.	/monitor/tv/X	GET
Update a SOAP/XML transaction monitoring policy.	/monitor/tv/X	POST
Replace a SOAP/XML transaction monitoring policy.	/monitor/tv/X	PUT
Delete a SOAP/XML transaction monitoring policy.	/monitor/tv/X	DELETE

Organizations

Action	URI	Method
View/search/filter the list of organizations.	/organization	GET
Create an organization.	/organization	POST
View the properties of an organization.	/organization/X	GET
Update the properties of an organization.	/organization/X	POST
Replace an organization.	/organization/X	PUT
Delete an organization.	/organization/X	DELETE
View/search/filter the list of logs associated with an organization.	/organization/X/log/	GET
View a log message associated with an organization.	/organization/X/log/X	GET
View/search/filter the list of notes associated with an organization.	/organization/X/note/	GET

Action	URI	Method
Add a note to an organization.	/organization/X/note/	POST
View a note associated with an organization.	/organization/X/note/X	GET
Update a note associated with an organization.	/organization/X/note/X	POST
Replace a note associated with an organization.	/organization/X/note/X	PUT
Delete a note associated with an organization.	/organization/X/note/X	DELETE
View/search/filter the list of files associated with an organization note.	/organization/X/note/X/media	GET
Get a media file associated with an organization note.	/organization/X/note/X/media/X	GET
Add a media file to an organization note.	/organization/X/note/X/media/X	PUT
View meta-data about a media file associated with an organization note.	/organization/X/note/X/media/X/info	GET

Performance Data

Action	URI	Method
View the index of available performance data resources.	/data_performance	GET
View the index of available performance data resources for devices.	/data_performance/device	GET
View normalized (rolled-up) data about availability and latency.	/data_performance/device/avail	GET
View normalized (rolled-up) data from one or more Dynamic Applications.	/data_performance/device/dynamic_app	GET
View normalized (rolled-up) data from file system usage policies.	/data_performance/device/filesystem	GET
View normalized (rolled-up) data from web content monitoring policies.	/data_performance/device/monitor_cv	GET

Action	URI	Method
View normalized (rolled-up) data from DNS monitoring policies.	/data_performance/device/monitor_dns	GET
View normalized (rolled-up) data from email round-trip monitoring policies.	/data_performance/device/monitor_email	GET
View normalized (rolled-up) data from a port monitoring policies.	/data_performance/device/monitor_port	GET
View normalized (rolled-up) data from system process monitoring policies.	/data_performance/device/monitor_process	GET
View normalized (rolled-up) data from Windows service monitoring policies.	/data_performance/device/monitor_service	GET
View normalized (rolled-up) data from SOAP/XML transaction monitoring policies.	/data_performance/device/monitor_tv	GET
View normalized (rolled-up) data about interface utilization.	/data_performance/interface	GET
View normalized (rolled-up) data about CBQoS objects.	/data_performance/interface	GET
View the index of available raw performance data resources.	/data_performance_raw	GET
View the index of available raw performance data resources for devices.	/data_performance_raw/device	GET
View raw data about availability and latency.	/data_performance_raw/device/avail	GET
View raw data from one or more Dynamic Applications.	/data_performance_raw/device/dynamic_app	GET
View raw data from file system usage policies.	/data_performance_raw/device/filesystem	GET
View raw data from web content monitoring policies.	/data_performance_raw/device/monitor_cv	GET
View raw data from DNS monitoring policies.	/data_performance_raw/device/monitor_dns	GET
View raw data from email round-trip monitoring policies.	/data_performance_raw/device/monitor_email	GET
View raw data from a port monitoring policies.	/data_performance_raw/device/monitor_port	GET

Action	URI	Method
View raw data from system process monitoring policies.	/data_performance_raw/device/monitor_process	GET
View raw data from Windows service monitoring policies.	/data_performance_raw/device/monitor_service	GET
View raw data from SOAP/XML transaction monitoring policies.	/data_performance_raw/device/monitor_tv	GET
View raw data about interface utilization.	/data_performance_raw/interface	GET
View raw data about CBQoS objects.	/data_performance_raw/cbqos	GET

PowerPacks

Action	URI	Method
View/search/filter the list of PowerPacks.	/powerpack	GET
View a PowerPack.	/powerpack/X	GET
Install a PowerPack.	/powerpack	Post a /filestore/powerpack resource.

Product SKUs

Action	URI	Method
View/search/filter the list of Product SKUs.	/product	GET
Create a new Product SKU.	/product	POST
View a Product SKU.	/product/X	GET
Update a Product SKU.	/product/X	POST
Replace a Product SKU.	/product/X	PUT
Delete a Product SKU.	/product/X	DELETE

Scale Values

Action	URI	Method
View/search/filter the list of scale values associated with metrics.	/scale	GET
View details about a scale value associated with metrics.	/scale/X	GET

Schedules

Action	URI	Method
View a list of schedules.	/schedule/	GET
Create a new schedule.	/schedule/	POST
View a schedule.	/schedule/X	GET
Update a schedule.	/schedule/X	POST
Delete a schedule.	/schedule/X	DELETE
View a list of tasks aligned to the schedule.	/schedule/X/task/X	GET

Streamer Push Proxy

Action	URI	Method
Return the current proxy configuration information.	/streamerpush/proxy	GET
Set the proxy information.	/streamerpush/proxy	POST
Toggle proxy on or off without deleting the configuration.	/streamerpush/proxy/toggle	POST

System Patches

Action	URI	Method
View/search/filter the list of patches registered in the system.	/system_patch	GET
View information about a registered patch.	/system_patch/X	GET
View/search/filter the list of log messages from the last execution of a patch.	/system_patch/X/log	GET
View a log message from the last execution of a patch.	/system_patch/X/log/X	GET
Register a patch file.	/system_patch	Post a /filestore/system_patch resource.
View/search/filter the list of staged patches.	/system_patch_stage	GET
View information about a staged patch.	/system_patch_stage/X	GET
Stage a patch file that has been registered in the system.	/system_patch_stage	Post a /system_patch resource.
View/search/filter the list of patches currently being installed.	/system_patch_deploy_active	GET
View information about a patch that is currently being installed.	/system_patch_deploy_active/X	GET
Install a staged patch.	/system_patch_deploy_active	Post a /system_patch_stage resource.

System Settings

Action	URI	Method
View the index of available system settings resources.	/system_settings	GET
View the global threshold settings.	/system_settings/system_thresholds	GET
Update the global threshold settings.	/system_settings/system_thresholds	POST

System Thresholds

Action	URI	Method
View/search/filter the list of system-level thresholds for metrics associated with interfaces.	/system_threshold	GET
View a system-level threshold for a metric associated with interfaces.	/system_threshold/X	GET
Update the settings for a system-level interface metric threshold.	/system_threshold/X	POST

Tasks

Action	URI	Method
View a list of tasks. A task is any item that can be scheduled, such as a discovery session.	/task/	GET
Create a new task.	/task/	POST
View a task.	/task/X	GET
Update a task.	/task/X	POST
Delete a task.	/task/X	DELETE

Action	URI	Method
View a list of schedules aligned to the task.	/task/X/schedule/X	GET

Themes

Action	URI	Method
View/search/filter the list of themes.	/theme	GET
Create a new theme.	/theme	POST
View a theme.	/theme/X	GET
Update a theme.	/theme/X	POST
Replace a theme.	/theme/X	PUT
Delete a theme.	/theme/X	DELETE

Threshold Overrides

Action	URI	Method
View/search/filter the list of threshold overrides that are in place for metrics associated with interfaces.	/threshold_value_override	GET
Add a threshold override for a metric on an interface.	/threshold_value_override	POST
View details about a threshold override for a metric associated with a specific interface.	/threshold_value_override/X	GET
Update a threshold override for a metric associated with a specific interface.	/threshold_value_override/X	POST
Replace a threshold override for a metric associated with a specific interface.	/threshold_value_override/X	PUT
Remove a threshold override for a metric associated with a specific interface.	/threshold_value_override/X	DELETE

Tickets

Action	URI	Method
View/search/filter the list of tickets.	/ticket	GET
Create a new ticket.	/ticket	POST
View the properties of a ticket.	/ticket/X	GET
Replace a ticket.	/ticket/X	PUT
Update a ticket.	/ticket/X	POST
View/search/filter the list of logs associated with a ticket.	/ticket/X/log/	GET
View a log message associated with a ticket.	/ticket/X/log/X	GET
View/search/filter the list of notes associated with a ticket.	/ticket/X/note/	GET
Add a note to a ticket.	/ticket/X/note/	POST
View a note associated with a ticket.	/ticket/X/note/X	GET
Update a note associated with a ticket.	/ticket/X/note/X	POST
Replace a note associated with a ticket.	/ticket/X/note/X	PUT
View/search/filter the list of files associated with a ticket note.	/ticket/X/note/X/media	GET
Get a media file associated with a ticket note.	/ticket/X/note/X/media/X	GET
Add a media file to a ticket note.	/ticket/X/note/X/media/X	PUT
View meta-data about a media file associated with a ticket note.	/ticket/X/note/X/media/X/info	GET
View/search/filter the list of external watchers associated with a ticket.	/ticket/X/watcher_ext	GET
Add an external watcher to a ticket.	/ticket/X/watcher_ext	POST
View an external watcher associated with a ticket.	/ticket/X/watcher_ext/X	GET
Update an external watcher associated with a ticket.	/ticket/X/watcher_ext/X	POST

Action	URI	Method
Replace an external watcher associated with a ticket.	/ticket/X/watcher_ext/X	PUT
Remove an external watcher from a ticket.	/ticket/X/watcher_ext/X	DELETE
View/search/filter the list of organization watchers associated with a ticket.	/ticket/X/watcher_org	GET
Add an organization watcher to a ticket.	/ticket/X/watcher_org	POST
View an organization watcher associated with a ticket.	/ticket/X/watcher_org/X	GET
Update an organization watcher associated with a ticket.	/ticket/X/watcher_org/X	POST
Replace an organization watcher associated with a ticket.	/ticket/X/watcher_org/X	PUT
Remove an organization watcher from a ticket.	/ticket/X/watcher_org/X	DELETE
View/search/filter the list of ticket queue watchers associated with a ticket.	/ticket/X/watcher_queue	GET
Add a ticket queue watcher to a ticket.	/ticket/X/watcher_queue	POST
View a ticket queue watcher associated with a ticket.	/ticket/X/watcher_queue/X	GET
Update a ticket queue watcher associated with a ticket.	/ticket/X/watcher_queue/X	POST
Replace a ticket queue watcher associated with a ticket.	/ticket/X/watcher_queue/X	PUT
Remove a ticket queue watcher from a ticket.	/ticket/X/watcher_queue/X	DELETE

Ticket Categories

Action	URI	Method
View/search/filter the list of ticket categories.	/ticket_category	GET
View the properties of a ticket category.	/ticket_category/X	GET

Ticket Chargeback

Action	URI	Method
View/search/filter the list of ticket chargeback entries.	/ticket_chargeback	GET
View the properties of a ticket chargeback entry.	/ticket_chargeback/X	GET

Ticket Logs

Action	URI	Method
View/search/filter the list of all ticket logs.	/ticket_log	GET
View a log message associated with a ticket.	/ticket_log/X	GET

Ticket Notes

Action	URI	Method
View/search/filter the list of all ticket notes.	/ticket_note	GET
View the properties of a ticket note.	/ticket_note/X	GET
Update a ticket note.	/ticket_note/X	POST
Replace a ticket note.	/ticket_note/X	PUT
View/search/filter the list of files associated with a ticket note.	/ticket_note/X/media	GET
Get a media file associated with a ticket note.	/ticket_note/X/media/X	GET
Add a media file to a ticket note.	/ticket_note/X/media/X	PUT

Action	URI	Method
View meta-data about a media file associated with a ticket note.	/ticket_note/X/media/X/info	GET

Ticket Queues

Action	URI	Method
View/search/filter the list of ticket queues.	/ticket_queue	GET
Create a new ticket queue.	/ticket_queue	POST
View the properties of a ticket queue.	/ticket_queue/X	GET
Update a ticket queue.	/ticket_queue/X	POST
Replace a ticket queue.	/ticket_queue/X	PUT
Delete a ticket queue.	/ticket_queue/X	DELETE

Ticket States

Action	URI	Method
View/search/filter the list of ticket states.	/ticket_state	GET
Create a new ticket state.	/ticket_state	POST
View the properties of a ticket state.	/ticket_state/X	GET
Update a ticket state.	/ticket_state/X	POST
Replace a ticket state.	/ticket_state/X	PUT
Delete a ticket state.	/ticket_state/X	DELETE

Unit Values

Action	URI	Method
View/search/filter the list of unit values associated with metrics.	/unit	GET
View details about a unit value associated with metrics.	/unit/X	GET

User Policies

Action	URI	Method
View/search/filter the list of user policies.	/account_policy	GET
Create a new user policy.	/account_policy	POST
View the properties of a user policy.	/account_policy/X	GET
Update the properties of a user policy.	/account_policy/X	POST
Replace a user policy.	/account_policy/X	PUT
Delete a user policy.	/account_policy/X	DELETE

Vendors

Action	URI	Method
View/search/filter the list of vendor records.	/vendor	GET
Create a new vendor record.	/vendor	POST
View a vendor record.	/vendor/X	GET
Update a vendor record.	/vendor/X	POST
Replace a vendor record.	/vendor/X	PUT

Action	URI	Method
Delete a vendor record.	/vendor/X	DELETE
View/search/filter the list of notes associated with a vendor record.	/vendor/X/note	GET
Add a note to a vendor record.	/vendor/X/note	POST
View a note associated with a vendor record.	/vendor/X/note/X	GET
Update a note associated with a vendor record.	/vendor/X/note/X	POST
Replace a note associated with a vendor record.	/vendor/X/note/X	PUT
View/search/filter the list of files associated with a vendor record note.	/vendor/X/note/X/media	GET
Get a media file associated with a vendor record note.	/vendor/X/note/X/media/X	GET
Add a media file to a vendor record note.	/vendor/X/note/X/media/X	PUT
View meta-data about a media file associated with a vendor record note.	/vendor/X/note/X/media/X/info	GET

© 2003 - 2026, ScienceLogic, Inc.

All rights reserved.

ScienceLogic™, the ScienceLogic logo, and ScienceLogic's product and service names are trademarks or service marks of ScienceLogic, Inc. and its affiliates. Use of ScienceLogic's trademarks or service marks without permission is prohibited.

ALL INFORMATION AVAILABLE IN THIS GUIDE IS PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED. SCIENCELOGIC™ AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT.

Although ScienceLogic™ has attempted to provide accurate information herein, the information provided in this document may contain inadvertent technical inaccuracies or typographical errors, and ScienceLogic™ assumes no responsibility for the accuracy of the information. Information may be changed or updated without notice. ScienceLogic™ may also make improvements and / or changes in the products or services described herein at any time without notice.



800-SCI-LOGIC (1-800-724-5644)

International: +1-703-354-1010